

MULTI-FAMILY DYNAMIC LOT SIZING
PROBLEMS
WITH COORDINATED REPLENISHMENTS

By

HASAN MURAT MERCAN

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1990

Copyright 1990

by

Hasan Murat Mercan

In The Name of Allah, All-Knowing, All-Wise

This work is dedicated to my parents Ummahan and Y. Nafiz Mercan
with full respect and love

ACKNOWLEDGEMENTS

I would like to express my gratitude to the members of my committee, S. Selcuk Erenguc, Harold P. Benson, Gary Koehler and Chun Y. Lee. I am particularly indebted to S. Selcuk Erenguc, my dissertation advisor, for his guidance, support and motivation.

My special thanks go to my wife, Inci, for her support, understanding, encouragement and lonely nights. My mother, Ummahan and my father Y. Nafiz Mercan are the primary vehicles of my life in general and my educational progress in particular. Nothing that I may do for them can pay back the benefits that I get with their efforts and guidance. I am therefore, grateful to my parents with an hope that this work is a slim contribution to their final accounts.

TABLE OF CONTENTS

	<u>page</u>
ACKNOWLEDGEMENTS.....	iv
ABSTRACT.....	vii
CHAPTERS	
1. INTRODUCTION.....	1
1.1. An Overview of Multi-Family Dynamic Lot Sizing Problems with Coordinated Replenishments (MFDLC).....	1
1.2. Statement of the Problem.....	4
1.2.1. Notation and Problem Formulation..	4
1.2.2. Some Properties of MFDLC.....	9
1.3. Organization of the Dissertation.....	13
2. A LITERATURE SURVEY OF THE RELATED RESEARCH....	16
2.1. Literature Review on MFDLP.....	23
2.2. Literature Review on MICDL.....	26
2.3. Literature Review on MICDLZ.....	29
2.3.1. Heuristic Procedures Based on Greedy Approach.....	29
2.3.2. Mathematical Programming Based Heuristic Procedures	35
2.4. Literature Review on Reverse Convex Programming.....	37
3. BRANCH AND BOUND ALGORITHMS FOR SOLVING MFDLC AND MFDLZ.....	39
3.1. Description of the Algorithm.....	40
3.2. Construction of Lower Bounding Problems..	43
3.2.1. Construction of Linear Underestimator.....	44
3.2.2. Construction of the Convex Envelope.....	49
3.2.3. Lower Bounding Subproblems.....	61
3.3. The Branching Process.....	65
3.4. A Formal Statement of the Branch and Bound Algorithm.....	68
3.5. Main Properties of the Algorithm.....	69

3.6. A Modified Branch and Bound Algorithm for MFDLZ.....	74
3.7. Preprocessing.....	75
4. HEURISTIC PROCEDURES FOR MFDLC AND MFDLZ.....	78
4.1. Description of the Heuristic Procedure.....	79
4.1.1. Individual Item Release Scheme....	86
4.1.2. Family Release Scheme.....	90
4.2. Finding a Starting Solution.....	94
4.3. Elimination of Inferior Alternatives in the Heuristic Procedure for MFDLZ.....	95
4.4. Feasibility in the Heuristic Procedures..	97
5. COMPUTATIONAL STUDY.....	103
5.1. Preprocessing the Data.....	104
5.2. Other Computational Issues.....	105
5.3. Computational Experience with the Branch and Bound Algorithm for MFDLZ.....	108
5.3.1. Experiment One.....	110
5.3.2. Experiment Two.....	121
5.3.3. Experiment Three.....	123
5.4. Computational Experience with the Branch and Bound Algorithm for MFDLC.....	126
5.5. Computational Experience with the Heuristic Procedure for MFDLZ.....	131
5.5. Computational Experience with the Heuristic Procedure for MFDLC.....	137
6. SUMMARY EXTENSIONS AND FURTHER RESEARCH.....	142
REFERENCES.....	147
BIOGRAPHICAL SKETCH.....	154

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

MULTI-FAMILY DYNAMIC LOT SIZING
PROBLEMS
WITH COORDINATED REPLENISHMENTS

By

HASAN MURAT MERCAN

May 1990

Chairman: S. Selcuk Erenguc

Major Department: Decision and Information Sciences

In this dissertation we consider multi-item production planning problems where items are grouped together into families. A family setup time and/or setup cost is incurred when an item in a family is produced. Also, for each replenishment of a product an individual setup time and/or setup cost may be incurred. The objective is to determine the time-phased production schedule that minimizes the total cost under demand and capacity constraints. In this dissertation we develop finite branch and bound algorithms for solving the problems. In addition, we develop certain preprocessing schemes. Heuristic procedures for the problems are also

presented in the dissertation. We also report some computational results for the branch and bound algorithms and heuristic procedures.

The dissertation includes a brief survey of multi-item production planning problems in general, and a detailed survey of multi-item capacitated dynamic lot sizing problems. We also include a brief review of certain classes of global optimization techniques due to their relevance to our research. Further research issues are discussed in the last chapter of the dissertation.

CHAPTER 1 INTRODUCTION

1.1. An Overview of Multi-Family Dynamic Lot Sizing Problem With Coordinated Replenishments

Single level multi-item dynamic lot sizing problem with coordinated replenishments is the determination of lot sizes of several products over a planning horizon where the products share a common resource. In many manufacturing environments the products may be classified into families where the setup structure makes coordinated replenishments attractive. In this type of lot sizing problem, there is a family (major) setup cost and/or setup time when at least one item is produced in a family in a period. An individual (minor) setup cost and/or setup time may also be incurred when a product is produced in a period. The objective is to determine the time-phased production schedule that minimizes the total relevant costs under demand and capacity constraints. We will refer to this type of problem as "Multi-family Dynamic Lot Sizing Problem with Coordinated Replenishments" (MFDLC).

MFDLC is notable mainly for at least two reasons. First it is found in many manufacturing environments. Products are grouped together to take the advantage of their similarities

in manufacturing and design. That is, those items whose coordination results in cost savings and increased productivity are classified into one family. For example, one manufacturing environment where this kind of product grouping is seen, is the group technology environment (Groover, 1980).

Secondly, a good approach to MFDLC can be an insight in solving more general lot sizing problems. One such problem is the lot sizing of several items which share more than one resource.

Consider a T period planning horizon where there are r distinct products which are classified into $m \leq r$ mutually exclusive and jointly exhaustive families. A family (major) setup cost K_i is incurred and a family setup time S_i $i \in \{1, 2, \dots, m\}$ is expended for each period $t \in \{1, 2, \dots, T\}$ when a positive amount of any product j in family i is produced. Here, K_i and S_i are the changeover cost and changeover time, respectively, associated with converting the facility from the production of some other family to production within family i . An individual (minor) setup cost k_j and setup time s_j may also be incurred when item j (of family i) is produced in period t , $t \in \{1, 2, \dots, T\}$. Therefore, k_j and s_j are the minor setup cost and setup time, respectively, which are incurred when the facility is switched to produce item j from the production of some other item within the same family. The objective of MFDLC is to find the time-phased production schedule that minimizes the total costs under demand and

capacity constraints. The demand is assumed to be deterministic and time-varying. In general, in a production environment the setup cost consists of the labor cost due to the setup time and certain out-of-pocket costs such as those incurred on material wastage. However, in many situations (Pinto and Mabert, 1986; Spencer, 1980) a major portion, if not all, of the setup cost is due to the labor cost of the setup time. If one makes the increasingly popular assumption that labor costs are fixed in the short run (Pinto and Mabert, 1986; Thompson, 1983), then in those cases where all, or a major portion of the setup cost, is due to the setup time, one can treat the setup cost as a fixed expense in the short run. Therefore, in the short term production planning problem, setup cost ceases to be a relevant factor. We will refer to this type of problem as "Multi-family Dynamic Lot Sizing Problem with Zero Setup Costs" (MFDLZ). Note that MFDLZ is a special case of MFDLC in which $K_i = 0$ for all $i \in \{1, 2, \dots, m\}$, and $k_j = 0$ for all $j \in \{1, 2, \dots, r\}$.

In this study we develop a branch and bound procedure for solving MFDLC. The procedure is then modified to obtain an algorithm for MFDLZ. In addition, we develop a heuristic procedure which is used to obtain an initial feasible solution and an upper bound for the optimal value of MFDLZ. This upper bound is used to preprocess the problem data.

1.2. Statement of the Problem

In this section we formulate MFDLC and give some properties of MFDLC that enable us to develop a branch and bound algorithm.

1.2.1. Notation and Problem Formulation

We will use the following notation throughout the dissertation.

T : number of periods in the planning horizon

r : number of products

I : index set of all products, $I = \{1, 2, \dots, r\}$

m : number of families

I_i : index set of products in family i , $i \in \{1, 2, \dots, m\}$.

For any $i \neq j$, $i, j \in \{1, 2, \dots, m\}$, $I_i \cap I_j = \emptyset$ and $I_1 \cup I_2 \cup \dots \cup I_m = I$

I_t^i : index set of products in family i in period t

r_i : cardinality of I_i , $\sum_{i=1}^m r_i = r$

For each $t \in \{1, 2, \dots, T\}$ and each $j \in I$ and $i \in \{1, 2, \dots, m\}$, we define the following.

x_{jt} : number of units of product j produced in period t

y_{jt} : ending inventory of product j in period t

y_{j0} : beginning inventory of product j in period 1

d_{jt} : demand for product j in period t

U_{jt} : upper bound on x_{jt}

c_j : variable production cost per unit of product j

h_j : inventory holding cost per unit per period for product j

C_t : available capacity in period t

a_j : capacity absorption rate per unit of product j

S_i : family setup time incurred if a positive amount of any product $j \in I_i$ is produced, $S_i > 0$

s_j : individual setup time incurred if a positive amount of product j is produced, $s_j \geq 0$

K_i : family setup cost incurred if a positive amount of any product $j \in I_i$ is produced, $K_i \geq 0$

k_j : individual setup cost incurred if a positive amount of any product j is produced, $s_j \geq 0$

Also let $n = 2r$, $\beta = 2rT$, $\alpha = rT$, $\beta_i = 2r_iT$ and $R_+^t = \{u \in R^t : u \geq 0\}$. To define the family setup time function, G_{it} , and the family setup cost function, F_{it} , for family i in period t , for each $i \in \{1, 2, \dots, m\}$ and each $t \in \{1, 2, \dots, T\}$, let $x_t^i = \{x_{jt} : j \in I_i\}$ and $y_t^i = \{y_{jt} : j \in I_i\}$. Then for each $i \in \{1, 2, \dots, m\}$, $t \in \{1, 2, \dots, T\}$ and $x_t^i \in R_+^{r_i}$, G_{it} is given by

$$G_{it}(x_t^i) = \begin{cases} 0 & \text{if } \sum_{j \in I_i} x_{jt} = 0 \\ S_i & \text{if } \sum_{j \in I_i} x_{jt} > 0 \end{cases} \quad (1)$$

and F_{it} is given by

$$F_{it}(x_t^i) = \begin{cases} 0 & \text{if } \sum_{j \in I_i} x_{jt} = 0 \\ K_i & \text{if } \sum_{j \in I_i} x_{jt} > 0 \end{cases} \quad (2)$$

For each $j \in I$, $t \in \{1, 2, \dots, T\}$ and $x_{jt} \in \mathbb{R}_+$, the capacity absorption function V_{jt} is defined as:

$$V_{jt}(x_{jt}) = \begin{cases} 0 & \text{if } x_{jt} = 0 \\ s_j + a_j x_{jt} & \text{if } x_{jt} > 0 \end{cases} \quad (3)$$

For each $j \in I$ and $t \in \{1, 2, \dots, T\}$ and $x_{jt} \in \mathbb{R}_+$, the production cost function $M_{jt}(x_{jt})$ is defined as:

$$M_{jt}(x_{jt}) = \begin{cases} 0 & \text{if } x_{jt} = 0 \\ k_j + c_j x_{jt} & \text{if } x_{jt} > 0 \end{cases} \quad (4)$$

Let $(x, y) = \{(x_{jt}, y_{jt}) : j = 1, 2, \dots, r ; t = 1, 2, \dots, T\}$ and for each $x^i_t \in \mathbb{R}^{r_i}_+$ let

$$\psi_{it}(x^i_t) = [F_{it}(x^i_t) + \sum_{j \in I_i} M_{jt}(x_{jt})]$$

and

$$\tau_{it}(x^i_t) = [G_{it}(x^i_t) + \sum_{j \in I_i} V_{jt}(x_{jt})]$$

Then for each $x \in \mathbb{R}^\alpha_+$,

$$\Psi(x) = \sum_{t=1}^T \sum_{i=1}^m \psi_{it}(x^i_t)$$

and for each $t \in \{1, 2, \dots, T\}$ and $x \in \mathbb{R}^\alpha_+$,

$$\tau_t(x) = \sum_{i=1}^m \tau_{it}(x^i_t)$$

Using the notation and definitions given above, MFDLC is stated as (P_1) .

$$(P_1) \quad \text{minimize } f(x, y) = \sum_{t=1}^T \sum_{j=1}^r h_j y_{jt} + \psi(x)$$

subject to:

$$y_{jt-1} + x_{jt} - y_{jt} = d_{jt} \quad \forall_{j,t} \quad (5)$$

$$\tau_t(x) \leq C_t \quad \forall_t \quad (6)$$

$$0 \leq x_{jt} \leq U_{jt} \quad \forall_{j,t} \quad (7)$$

$$y_{jt} \geq 0 \quad \forall_{j,t} \quad (8)$$

$$y_{jT} = 0 \quad \forall_j \quad (9)$$

$$y_{j0} = 0 \quad \forall_j \quad (10)$$

In MFDLZ it is assumed that setup costs are negligible. Therefore the term $\psi(x)$ drops from the objective function of (P_1) . We now state MFDLZ problem as

$$(PZ_1) \quad \text{minimize } g(x, y)$$

subject to (5), (6), (7), (8), (9), (10)

where

$$g(x, y) = \sum_{j=1}^m \sum_{t=1}^T h_j y_{jt}$$

We note that if the upper bounds of constraints (7) are not explicitly available, these can be easily obtained either from constraints (5), (9) and (10) or from constraint (6). We make the following additional assumptions.

- A. There can be no backlogging, demands should be satisfied in their respective periods of occurrence.
- B. Replenishment lead times are negligible.

C. Inventories at the beginning and at the end of the horizon are zero.

D. A setup for product $j \in I$ is needed in period t , whether or not this product was produced in period $t-1$.

With assumption C we can drop y_{j0} and, therefore, constraint (10) from the problem. Hence the problem is one of determining a vector $(x^*, y^*) = (x_{11}^*, x_{12}^*, \dots, x_{rT}^*, y_{11}^*, y_{12}^*, \dots, y_{rT}^*)$, if it exists, which satisfies $f(x^*, y^*) = f^* = \text{minimize } f(x, y)$ subject to (5)-(9). We also note that the parameters $h_j, c_j, k_j, s_j, a_j, S_i, K_i$ could be made time-varying. Nonetheless, this does not really change the problem formulation. Moreover, these parameters are expected to be constant over the short run. Since all demand must be satisfied and production cost is constant over time, we can drop the variable term $c_j x_{jt}$ from the function $M_{jt}(x_{jt})$.

Note that MFDLC can be formulated as a mixed integer linear programming problem and can be solved with a mixed integer software package. The number of constraints and the number of variables in this formulation, however, are much more than those in (P_i) . Therefore, this may be a computationally inefficient approach. For the purpose of further discussions we need to give the mixed integer formulation for MFDLC.

Let $(x, y, w, q) = \{(x_{jt}, y_{jt}, w_{it}, q_{jt}) : i = 1, 2, \dots, r; j \in I_i; t = 1, 2, \dots, T\}$. Following is a mixed integer linear

programming formulation (IP₁) of MFDLC.

$$(IP_1) \text{ minimize } z(x, y, w, q) = \sum_{t=1}^T \left[\sum_{i=1}^m [K_i q_{it} + (\sum_{j \in I_i} k_j w_{jt} + \sum_{j \in I_i} h_j y_{jt})] \right]$$

subject to

$$y_{jt-1} + x_{jt} - y_{jt} = d_{jt} \quad \forall_{j,t} \quad (11)$$

$$\sum_{i=1}^m [S_i q_{it} + \sum_{j \in I_i} (s_j w_{jt} + a_j x_{jt})] \leq c_t \quad \forall_t \quad (12)$$

$$0 \leq x_{jt} \leq w_{jt} U_{jt} \quad \forall_{j,t} \quad (13)$$

$$0 \leq \sum_{j \in I_i} x_{jt} \leq q_{it} \sum_{j \in I_i} U_{jt} \quad \forall_{i,t} \quad (14)$$

$$y_{jt} \geq 0 \quad \forall_{j,t} \quad (15)$$

$$y_{jT} \geq 0 \quad \forall_j \quad (16)$$

$$q_{it} \in \{0, 1\} \quad \forall_{i,t} \quad (17)$$

$$w_{jt} \in \{0, 1\} \quad \forall_{j,t} \quad (18)$$

Note that (IP₁) has rT additional constraints (constraints (14)) and T(r+m) additional 0-1 variables over (P₁).

1.2.2. Some Properties of MFDLC

It has been shown (Erenguc and Benson, 1986) that for each $i \in \{1, 2, \dots, m\}$ and each $t \in \{1, 2, \dots, T\}$, G_{it} and F_{it} are concave and lower semicontinuous on R^r_{i+} . Furthermore, it is a well-known fact that for each $j \in I$ and each $t \in \{1, 2, \dots, T\}$, V_{jt} and M_{jt} are concave and lower semicontinuous on R_+ . Therefore, for each $i \in \{1, 2, \dots, m\}$, for each $t \in \{1, 2, \dots, T\}$ and for any $x^i_t \in R^r_{i+}$, $\psi_{it}(x^i_t)$ and $r_{it}(x^i_t)$ are concave and lower

semicontinuous functions (Magnasarian, 1969). Consequently, for each $t \in \{1, 2, \dots, T\}$ and for any $x \in R_+^n$, $r_t(x)$ and $\psi(x)$ are concave and lower semicontinuous functions. A constraint $g(x) \leq 0$ is said to be a **reverse convex constraint**, if for any $x \in R_+^n$, $g(x)$ is a concave function. Therefore, (P_1) involves a minimization of a concave objective function over the intersection of a bounded polyhedral set and T reverse convex constraints. There are a few general purpose approaches for solving global optimization problems involving reverse convex constraints all of which have infinite convergence. As we will discuss in the following sections due to the special structure of the reverse convex constraints (constraint (6) in (P_1)) and the special structure of the objective function, we are able to develop a finite branch and bound algorithm for solving (P_1) .

Property 1. The constraint set of (P_1) is compact.

Proof. First we note that constraints (5), (7), (8) and (9) form a bounded polyhedron in R_+^n . It follows from the lower semicontinuity of G_{it} and V_{jt} for each $i \in \{1, 2, \dots, m\}$, $j \in I$ and $t \in \{1, 2, \dots, T\}$ that constraint (6) form a closed set on R_+^n . Then the intersection of the set formed by the constraints (5), (7), (8), (9) and those formed by the constraint (6) is compact on R_+^n . ■

For each $i \in \{1, 2, \dots, m\}$ and each $t \in \{1, 2, \dots, T\}$ we define the following sets

$$A_t^i = \{x_{jt} > 0: j \in I_i\}$$

$$I_{tA}^i = \{j: x_{jt} \in A_t^i\}$$

Note that $A_t^i \subseteq x_t^i$ and $I_{tA}^i \subseteq I_t^i$. For any set I_{tA}^i and I_{tB}^i let $I_{tA}^i \setminus I_{tB}^i = \{j \in I_{tA}^i: j \notin I_{tB}^i\}$. We now develop some properties for $\psi_{it}(x_t^i)$ and $\tau_{it}(x_t^i)$ which enable us to utilize some of the results of Erenguc and Benson (1986).

Property 2. For each $i \in \{1, 2, \dots, m\}$ and for each $t \in \{1, 2, \dots, T\}$, and for any A_t^i the following statements are true

- a) $\psi_{it}(\emptyset) = 0$
- b) $\tau_{it}(\emptyset) = 0$
- c) If $A_t^i \neq \emptyset$, then $\psi_{it}(A_t^i) \geq 0$
- d) If $A_t^i \neq \emptyset$, then $\tau_{it}(A_t^i) > 0$

Proof. a) This follows from the fact that $F_{it}(x_t^i) = 0$ when $A_t^i = \emptyset$ and $M_{jt}(x_{jt}) = 0$ when $x_{jt} = 0$. Hence $\psi_{it}(\emptyset) = 0$.

b) This follows a similar line of reasoning as (a)

c) Since $A_t^i \neq \emptyset$ and $K_i \geq 0$ there exists at least one $x_{jt} > 0$. Hence $F_{it}(x_t^i) = K_i \geq 0$. Also note that for all $j \in I_i$ $c_j, k_j \geq 0$ therefore, $M_{jt}(x_{jt}) \geq 0$. The proof follows from the fact that,

$$\psi_{it}(A_t^i) = F_{it}(A_t^i) + \sum_{j \in I_{tA}^i} k_j + c_j x_{jt} \geq 0.$$

d) Since $S_i > 0$, $s_j \geq 0$ and $a_j \geq 0$ the proof follows the same line of proof as part c.

Property 3 Suppose $B_t^i \subseteq A_t^i$, $i \in \{1, 2, \dots, m\}$, $t \in \{1, 2, \dots, T\}$. For each $i \in \{1, 2, \dots, m\}$ and for each $t \in \{1, 2, \dots, T\}$,

- a) $\psi_{it}(B_t^i) \leq \psi_{it}(A_t^i)$ and

b) $\tau_{it}(B_t^i) \leq \tau_{it}(A_t^i)$ are true.

Proof. Suppose $A_t^i = \emptyset$, then the proofs of part a and part b follow as a direct consequence of property 2a and 2b, respectively.

Suppose $A_t^i \neq \emptyset$, but $B_t^i = \emptyset$. The proofs of part a and part b follow as a direct consequence of property 2a and 2c, and 2b and 2d, respectively. So for the rest of the proof we will assume that $A_t^i \neq \emptyset$ and $B_t^i \neq \emptyset$.

$$\begin{aligned} \text{a) } \psi_{it}(A_t^i) &= K_i + \sum_{j \in I_{tA}^i} K_j + c_j x_{jt} \\ &= K_i + \sum_{j \in I_{tB}^i} (K_j + c_j x_{jt}) + \sum_{j \in I_{tA}^i \setminus I_{tB}^i} (K_j + c_j x_{jt}) \\ &= \psi_{it}(B_t^i) + \sum_{j \in I_{tA}^i \setminus I_{tB}^i} (K_j + c_j x_{jt}) \end{aligned}$$

Since $\sum_{j \in I_{tA}^i \setminus I_{tB}^i} (K_j + c_j x_{jt}) \geq 0$, $\psi_{it}(B_t^i) \leq \psi_{it}(A_t^i)$.

b) This follows a similar line of reasoning as part (a).

Property 4. Suppose $C_t^i \subseteq B_t^i \subseteq A_t^i$, $i \in \{1, 2, \dots, m\}$, $t \in \{1, 2, \dots, T\}$. For each $i \in \{1, 2, \dots, m\}$ and for each $t \in \{1, 2, \dots, T\}$ the following statements are true

$$\begin{aligned} \text{a) } \psi_{it}(A_t^i) - \psi_{it}(B_t^i) &= \psi_{it}(A_t^i \setminus C_t^i) - \psi_{it}(B_t^i \setminus C_t^i) \\ \text{b) } \tau_{it}(A_t^i) - \tau_{it}(B_t^i) &= \tau_{it}(A_t^i \setminus C_t^i) - \tau_{it}(B_t^i \setminus C_t^i). \end{aligned}$$

Proof. Since the proof of (b) is exactly the same as the proof of (a) we will only prove (a).

If $A_t^i = \emptyset$, then the proof is trivial. If $A_t^i \neq \emptyset$ but $B_t^i = \emptyset$, then $C_t^i = \emptyset$ which implies

$\psi_{it}(A_t^i) - \psi_{it}(B_t^i) = \psi_{it}(A_t^i \setminus C_t^i) - \psi_{it}(B_t^i \setminus C_t^i) = \psi_{it}(A_t^i)$. The proof then follows as a direct consequence of property 2a. Also the

case where $A_t^i \neq \emptyset$, $B_t^i \neq \emptyset$ and $C_t^i = \emptyset$ is trivial. So, for the rest of the proof we will assume that $A_t^i \neq \emptyset$, $B_t^i \neq \emptyset$ and $C_t^i \neq \emptyset$.

If $A_t^i \setminus C_t^i = \emptyset$, then the proof is trivial because $A_t^i \setminus C_t^i = \emptyset$ implies that $A_t^i = B_t^i = C_t^i$. If $A_t^i \setminus C_t^i \neq \emptyset$ and $B_t^i \setminus C_t^i = \emptyset$, then $B_t^i = C_t^i$. Therefore, $\psi_{it}(A_t^i \setminus C_t^i) - \psi_{it}(B_t^i \setminus C_t^i) = \psi_{it}(A_t^i \setminus B_t^i) - K_i$. The proof follows since $\psi_{it}(A_t^i) - \psi_{it}(B_t^i) = \psi_{it}(A_t^i \setminus B_t^i) - K_i$. So, for the rest of the proof we will also assume that $A_t^i \setminus C_t^i \neq \emptyset$ and $B_t^i \setminus C_t^i \neq \emptyset$.

$$\begin{aligned} \psi_{it}(A_t^i \setminus C_t^i) - \psi_{it}(B_t^i \setminus C_t^i) &= \sum_{j \in I_{tA}^i \setminus I_{tB}^i} [k_j + C_j x_{jt}] - \sum_{j \in I_{tB}^i \setminus I_{tC}^i} [k_j + C_j x_{jt}] + \sum_{j \in I_{tB}^i \setminus I_{tC}^i} [k_j + C_j x_{jt}] \\ &= \sum_{j \in I_{tA}^i \setminus I_{tB}^i} [k_j + C_j x_{jt}] \end{aligned}$$

The proof follows since

$$\psi_{it}(A_t^i) - \psi_{it}(B_t^i) = \psi_{it}(A_t^i \setminus B_t^i) - K_i = \sum_{j \in I_{tA}^i \setminus I_{tB}^i} [k_j + C_j x_{jt}]. \blacksquare$$

1.3. Organization of the Dissertation

MFDLC is one of the most general class of single level multi-item dynamic lot sizing problems. It is therefore important to bring some insight into this problem for its own sake. A procedure which is capable of solving MFDLC can also be used for solving other variants of the single level multi-item dynamic lot sizing problems.

In many manufacturing environments, a product is produced through a series of operations in several work centers. Although

a solution procedure for MFDLC will not in general be applicable to multi-item dynamic lot sizing problem with multiple work centers, it is an important step in developing solution procedures for lot sizing problems involving multiple work centers.

Florian, Lenstra and Rinnooy Kan (1980) proved that even special case of MFDLC is NP-hard problem. It is therefore very difficult to solve realistic problems in MFDLC optimally. As a result of this, the literature seems to be developing mainly in the direction of effective and computationally feasible heuristic procedures for MFDLC and its variants. Developing a procedure that solves MFDLC optimally is still important, for the characteristics of an optimal solution to the problem can be studied to guide the development of good heuristic procedures. In addition, a solution procedure is important for generating optimally solved benchmark problems against which heuristic procedures can be tested.

In this dissertation we develop branch and bound algorithms for MFDLC and MFDLZ. We attempt to develop preprocessing procedures for the problem data to reduce the computational effort. We also develop heuristic procedures for MFDLC and MFDLZ. Finally, we report some computational results and discuss the possible extensions of the research.

We believe that the results of this study will bring some insight to the researchers in this field. In addition, some of the results of this study will be applicable to certain manufacturing environments.

A literature survey of related literature is given in chapter 2. Chapter 3 of this dissertation is devoted to a branch and bound algorithm for MFDLC and MFDLZ. In this chapter we also discuss how the problem data for MFDLC and MFDLZ can be preprocessed. In chapter 4 we develop a heuristic procedure for MFDLC and MFDLZ. Computational results for the branch and bound algorithms and heuristic procedures are reported in chapter 5. Conclusion and further research issues are discussed in chapter 6.

CHAPTER 2

A LITERATURE SURVEY OF RELATED RESEARCH

Inventory control problems are one of the most extensively studied application areas for Operations Research/Management Sciences. Several reviews on the subject have been published (see Whitin, 1954; Hanssmann, 1961; Veinot, 1966; Clark, 1972; Aggarwal, 1974; Nahmias, 1978; Silver, 1981; Berry and Mabert, 1981; Gelders and Van Wassenhove, 1981; Aksoy and Erenguc, 1988; Bahl, Ritzman and Gupta, 1987). Although many algorithms and solution procedures have been developed for inventory control models there still remain problems to be solved. Since our interest in this research is a multi-item capacitated lot sizing problem which is a special class of inventory models (for a general classification of inventory models see for example Bahl, Ritzman and Gupta, 1987; Peterson and Silver, 1979; Johnson and Montgomery, 1974), we will only review the pertinent literature.

The existing solution procedures for multi-item lot sizing problems fall into one of the following three categories:

1. Treating each item independently without considering the coordination among them;
2. hierarchical approach; and
3. monolithic approach.

When items are treated independently family and individual setup time and/or setup cost is incurred each time a product is produced, regardless of the production level of other products in that period. Also assuming independence among products implies the treatment of each product individually. Therefore, the problem becomes a single-item capacitated lot sizing problems (SICLP). Since SICLP is a special case of multi-item capacitated lot sizing problems, we will review SICLP under that category. In some single-item models capacity constraint is ignored yielding single-item uncapacitated lot sizing problem (SIULP). A dynamic programming type optimal solution procedure to SIULP was developed by Wagner and Whitin (1958). Wagner and Whitin (W-W) algorithm was criticized from practitioners point of view. Although W-W was a nonpolynomial algorithm, due to the available computer technologies, the procedure was then considered to be computationally impractical. The second criticism to W-W was that it was not easily comprehensible by the practitioners. Therefore, heuristic procedures were developed to solve SIULP. Most of these heuristic procedures were based on the Economic Order Quantity (EOQ) model which was developed by Harris (1915). Among these heuristic

procedures are Lot for Lot, Periodic Order Quantity, Part Period Balancing and Incremental Part Period Balancing algorithms (the reader is referred to Tersine, 1988 for a complete discussion of these heuristic procedures.) Silver and Meal (1973) developed a heuristic procedure for SIULP based on the cost per period criterion. We will briefly discuss W-W and Silver and Meal procedures in the following paragraphs due to their relevance to our research.

Wagner and Whitin (1958) developed a solution procedure for solving SIULP optimally. They show that at an optimal solution at any period either the production variable is positive and the inventory level in the beginning of that period is zero, or the production variable is zero and the inventory level is positive, or they both are zero and the demand for that period is also zero. This property is referred to as W-W property and it forms a basis for many of the heuristic procedures.

Silver and Meal (1973) developed a procedure that minimizes the cost per period of each successive order cycle. This procedure is relatively simple and furthermore it produces near optimal solutions. Many researchers henceforth applied the results of Silver and Meal (1973) to the multi-item capacitated dynamic lot-sizing models.

Although to assume independence among the products is computationally attractive, there are several advantages in coordinating the replenishments of related products. These

include savings in setup cost and time, savings on unit transportation and unit production costs. In addition to computational complexity, there are some disadvantages of using coordinated replenishment procedures such as increasing system control cost and/or average inventory level. Peterson and Silver (1979) discuss pros and cons of coordinated replenishment in detail.

Two distinct approaches for multi-item lot sizing problems with coordinated replenishments have appeared in the literature. The first approach is called a **monolithic approach** and the second approach is called a **hierarchical approach** (Graves, 1982). In the monolithic approach, the problem is formulated as a single model and solution procedures for this model are sought. We will review the solution procedures which fall under this category in sections 2.2 and 2.3. In the hierarchical approach, the problem is partitioned into a hierarchy of subproblems and solution procedures for these subproblems are developed. Since our approach is a monolithic approach, we will briefly review hierarchical approach in the following paragraph.

Hierarchical production planning (HPP) was first introduced by Hax and Meal (1975). Hax and Meal (1975) proposed the following aggregation of products: **items** are the end products; **product types** are groups of items having similar unit costs, holding costs, productivities and seasonalities; **families** are groups of items within a product

type that share similar setup costs and times. Whenever a facility is set to produce an item in a family, all other items in the same family can be produced with the incursion of minor setup costs and times. Production quantities for the items in the planning horizon are determined in three steps. First, the production quantities for the types are determined through **aggregate production planning**. These production quantities are then used in the determination of the production quantities of the families. This process is referred to as **family disaggregation** and it is mainly the distribution of the production quantity of a type among the families within that type. In the third step, the distribution of the production quantity of a family among the items within that family is done. This is called **item disaggregation**. A mathematical formulation to HPP was first provided by Bitran and Hax (1977). Bitran and Hax (1977) formulate the aggregate production planning problem as a regular knapsack problem. They formulate two different convex knapsack problems for solving family and item disaggregation problem. Bitran and Hax (1981) developed efficient solution procedures to solve these convex knapsack problems. Bitran, Haas and Hax (1981) modified these procedures. Graves (1982) formulated aggregate planning and family disaggregation process as a monolithic mixed integer problem and applied Lagrangean relaxation technique to solve this problem. For a detail discussion of HPP the reader is referred to Hax and Candea (1984).

Buffa (1983) defines capacity as the limiting capability of a productive unit to produce within the stated amount of time. Although it is possible to increase the capacity of the production unit to a certain extent (for example by overtime, an additional shift or hiring more workers) at a certain cost, capacity planning is a long term issue. Therefore, in the short run capacity of a production unit is not a policy variable. However, even in the short run the capacity might vary from one period to another due to the holidays and scheduled maintenance. Nevertheless, in most of the production plants, if not all, capacity imposes certain limitations on the output a production unit produces. Therefore, ignoring capacity in the lot sizing models is rather a restrictive assumption. The capacity is mainly used for the preparation of the unit for the production of items (which is referred to as setup) and the production of the items.

Just-In-Time (JIT) is a manufacturing system which was first introduced and implemented by the Japanese manufacturers. Buffa and Sarin (1987: p) defines JIT as "producing the necessary items in the quantities needed, at the time they are needed." Recent studies of the JIT system (Sugimori, Kusunoki and Cho, 1971; Krajewski, King, Ritzman and Wang, 1983; Ritzman, King and Krajewski, 1984) claim that setup times can be reduced to modest level. Hence these studies tend to assume negligible setups in their models. In

many manufacturing environment, however, there are limits on reducing setup times. When setup time is considered even a single item problem can become complex. Florian, Lenstra and Rinnooy Kan (1980) show that several versions of single level capacitated lot sizing problem are NP-hard. Bitran and Yanasse (1982) show that several single-item capacitated problems which could be solved by a polynomial time algorithm become NP-hard when a second item is introduced.

The setup structure studied in this research has received a considerable attention in the dynamic lot sizing literature. However, previous research on dynamic lot sizing models with family and individual setups introduced these setups only as a cost and assumed no capacity constraint. Therefore, the previous works on dynamic lot sizing model with family and individual setups are more suited for a procurement rather than a manufacturing environment. We will refer to this type of problem as "Multi-family dynamic lot sizing model with coordinated replenishments and infinite capacity" (MFDLP). Although these models assumed away the capacity constraints, the kind of capacity constraints and objective function we studied in this research were alluded to in these models. Therefore, we will briefly review the researches on MFDLP in Section 2.1.

Note that in the special case when $m=r$, MFDLC reduces to the so called multi-item capacitated dynamic lot sizing problem (MICDL). The literature on MICDL will be discussed in section 2.2.

Incurring setup time into the model brings additional difficulty to the problem, because the capacity constraints will then become nonlinear. To avoid this complication some of the researchers include the opportunity cost of using the capacity to the setup cost and henceforth get rid of the setup time in their model. We will refer to this type of problems as multi-item capacitated dynamic lot sizing problem with zero setup time (MICDLZ). We will review the studies on MICDLZ in Section 2.3.

As shown in Section 1.2 (P_1) involves a minimization of a concave objective function over the intersection of a bounded polyhedral set and T reverse convex constraints. We will review the general purpose approaches for solving global optimization problems involving reverse convex constraints in Section 2.4 because of their relevance to our problem.

2.1. Literature Review on MFDLP

Coordinated cost structure in the case of level demand has been treated by Goyal (1974), Brown (1967), Doll and Whybark (1973) and Silver (1976). The case of time varying demand and cost patterns with the coordinated cost structure has been dealt with Erenguc (1988), Kao (1979), Silver (1979), Veinot (1969) and Zangwill (1966). In all of these studies, a single family of products were assumed. The setup cost structure in these studies is exactly the same with MFDLP.

The results and the algorithms presented therein directly extend to the case of multiple families since the families are independent. In this case m independent multi product lot sizing problems will be solved. We will now briefly discuss the results and the algorithms proposed by Erenguc (1988), Kao (1979), Silver (1979), Veinot (1969), and Zangwill (1966).

Zangwill (1966) showed that there exists an optimal solution to MFDLP with the property that the production of any item in any period can take place if the beginning inventory of that item is zero in that period. This property is first established by Wagner and Whitin (1958) for a single-item uncapacitated dynamic lot sizing problem with concave costs. Based on this property Zangwill (1966) developed a dynamic programming formulation that uses time as the stage variable and the starting inventory levels of the products in a family as the state variable. In this formulation the size of the state space increases exponentially thereby making the procedure computationally infeasible. The total number of

states for each family i , $i \in \{1, 2, \dots, m\}$ is $\sum_{t=1}^T t^{r_i}$.

Kao (1979) presented an alternative dynamic programming formulation using an acyclic network which is then solved by finding the shortest path from the source node to the sink node in this network. In Kao's formulation each state corresponds to a regeneration point implied by the W-W property. Although the definition of "a state" results in a

smaller state space and fewer arcs in the shortest path network than those proposed by Zangwill (1966), the size of the state space is still a limiting factor in this formulation.

Silver (1979) considered a special case of this problem where he assumes constant inventory, procurement, and family and individual setup costs over time. His dynamic programming formulation is similar to that of Kao's formulation. He concluded that dynamic programming approaches are suitable only for relatively small values of T and r_i , $i \in \{1, 2, \dots, m\}$ and for larger values of T and r_i , $i \in \{1, 2, \dots, m\}$ heuristic procedures are needed.

Veinot (1969) proposed another approach to solve MFDLP. In any period either at least one product is produced or no product is produced at all. He considers all such possible patterns of production schedule. For each of these possible patterns the minimum total cost is found by using the W-W procedure. Then the cost associated with each pattern is compared and the least costly pattern is chosen as the optimal production schedule. Although Veinot's approach enumerates all possible production patterns, it appears to outperform dynamic programming type solution procedures.

Erenguc (1988) developed a branch and bound algorithm for solving MFDLP. He uses the algorithm given in Erenguc and Benson (1986) and the idea of possible patterns proposed by Veinot (1969). His algorithm, however, does not explicitly

enumerate all possible production patterns. Erenguc (1988) underestimates the objective function of the problem which becomes a separable concave function over appropriate domains. At each node of the branch and bound tree a subproblem which has the underestimating function as an objective function is solved. This subproblem turns out to decompose into r_i single item problems. Optimal solutions for the single item problems are easily obtained by using the W-W approach. Erenguc (1988) tested his algorithm against randomly generated problems. He reported reasonable CPU times for even a realistically sized problems. He could also solve all of the problems in a very reasonable CPU time that could not have been solved in a reasonable CPU time by the other procedures.

2.2. Literature Review on MICDL

MICDL was first introduced by Manne (1958). His formulation is slightly different than the formulations that followed. He allows limited overtime in his model and he assumes that inventory carrying costs are negligible. The problem is to minimize total overtime labor requirements under demand, regular time and overtime capacity constraints. Although this problem is different than MICDL, its feasible region is similar to that of MICDL. Therefore, some of the results found in this study are valid for MICDL. He formulates MICDL as 0-1 mixed integer problem. However, to

make the model computationally tractable he considers only the W-W type solutions that are feasible to MICDL. Using these solutions he constructs another 0-1 mixed integer program. The solution to the latter program yields an upper bound for the optimal value of MICDL. This problem is also computationally prohibitive. Instead of solving this problem he solves a linear programming relaxation of the problem which is obtained by relaxing the binary variables. He shows that in this relaxed problem at most T binary variables will have fractional values. If $r \gg T$ an optimal solution to an LP relaxation can be a good approximate solution to the original problem.

The efficiency of Manne's approach was tested by Dzielinski, Baker and Manne (1963). They showed that his LP relaxation generates reliable approximate solutions.

Bitran and Matsau (1986) reported some results on Manne's formulation. Let (IZ_1) be the 0-1 mixed integer programming formulation which only considers W-W solutions and (LZ_1) be its LP relaxation obtained by relaxing integrality constraints. Let D_{IP_1} , D_{IZ_1} and D_{LZ_1} be the feasible regions of problems (IP_1) , (IZ_1) and (LZ_1) , respectively. They show that $D_{IP_1} \subset D_{IZ_1} \subset D_{LZ_1}$. They also show that the difference between the optimal objective function value of (LZ_1) and the optimal objective function value of (IP_1) does not exceed the sum of all individual setup costs minus the minimum of the setup costs.

Although Manne's formulation can produce "good" approximate solution to MICDL it does not necessarily guarantee a feasible solution. Bitran and Matsau (1986) also show that in any period, $t \in \{1, 2, \dots, T\}$, capacity violation is bounded by R_t/C_t where R_t is the sum of the T largest s_j , $j \in I$.

In Manne's formulation 2^{T-1} W-W solutions can be generated. Therefore, the problem size grows exponentially as T increases. In the case of higher T values, solving (LZ_1) can become computationally prohibitive. Dzielinski and Gomory (1965) applied Dantzig Wolfe decomposition technique to (LZ_1) to lessen the computational burden. Lasdon and Terjung (1971), and Bahl (1983) applied column generation scheme for this purpose. Newson and Kleindorfer (1975) applied Lagrangean relaxation to MICDL and relaxed the capacity constraints. They used generalized duality theory to update the multipliers.

Solution procedures proposed by Manne (1958), Dzielinski and Gomory (1965), and Lasdon and Terjung (1971) do not always guarantee a feasible solution, and a good approximate solution is only assured when the number of products are much larger than the number of periods. Newson (1975) suggested a solution procedure without these limitations. He generates all W-W solutions for each product and ranks them in ascending order of their costs. Among these W-W solutions first minimum cost schedule for each item is selected. If infeasibility in some periods occurs, the schedule of one of the products is

replaced with the next minimum cost schedule and then feasibility of the revised schedule is checked. This procedure is repeated until feasibility for each period is maintained. Newson's procedure always generates a feasible solution to MICDL provided that there exists a W-W type feasible solution.

2.3. Literature Review On MICDLZ

Two major lines of researches can be seen in solving MICDLZ. One group of researchers has focused on developing heuristic procedures based on greedy approaches. The other group has exploited the mathematical structure of the problem either to develop a heuristic procedure or to solve MICDLZ optimally. We briefly discuss these studies in the following sections.

2.3.1. Heuristic Procedures Based on Greedy Approach

Eisunhut (1975) developed a heuristic procedure based on the heuristic procedure developed by Silver and Meal (1973). Starting from the first period and starting with the item which shows the greatest potential reduction in cost per period, production of items in the first period is increased until either the capacity in that period is exhausted or until no additional reduction in the cost per period is possible for

any product. The procedure then moves to the next period whose capacity is not fully consumed. The procedure is repeated until the end of the planning horizon.

Eisunhut's procedure moves forward. In such unidirectional procedures it is possible to end up with an infeasible solution events though the problem is feasible. Lambrecht and Vanderveken (1979) developed an algorithm that overcomes this drawback. Their procedure is known as "extended Eisunhut" procedure and it differs from Eisunhut's procedure in two ways. First they use a slightly different cost evaluation function which is based on the part-period balancing criterium. Second, after the procedure determines the lot sizes unidirectionally, infeasibility in any period is eliminated by shifting some of the production to an earlier period. The choice of shifting to maintain feasibility is mainly done by the relative magnitude of the holding cost that a shift might create. The extended Eisunhut procedure always generates a feasible solution provided that there exists a feasible solution to MICDLZ.

Dixon and Silver (1981) proposed a greedy heuristic procedure similar to Eisunhut's procedure, the difference being in the cost criterion that they used. Dixon and Silver (1981) develop the concept of marginal decrease in the average cost per unit time. Marginal decrease in the average cost is calculated as the difference of the cost of satisfying the demand of an item for periods $\{k, k+1, \dots, t\}$ in period k and

the cost of satisfying the demand of an item for periods $(k, k+1, \dots, t, t+1)$ in period k , for any $k \in \{1, 2, \dots, T\}$. To ensure feasibility they impose a constraint on the production level in each period. The amount of total production in a period which will be used to satisfy the requirements of the next periods must be greater or equal to the sum of the difference of total requirements and the capacity of these periods. Dixon and Silver (1981) also develop some additional steps in improving the solution of the greedy procedure. The following policies can be implemented provided that they improve the solution.

i. In case of an excess capacity in a period the production of an item in some other periods can be shifted to this period such that this item is not produced in those periods. This implies savings in the setup cost and a possible increase in the inventory carrying cost.

ii. Similarly, two lots which are produced in different periods can be merged into one lot in some other period which has sufficient capacity. Merging the two lots reduces the setup cost but increases the inventory carrying cost.

iii. The lots can be interchanged so that W-W property is maintained. This operation may result a reduction in the setup cost and an increase in the carrying cost or vice versa.

iv. The lot sizes of items can be increased in such a way that the solution is an extreme point of MICDLZ. This operation may also change the cost pattern as in (iii).

As shown by Dogramaci, Panayiotopoulos and Adam (1981) the constraint developed by Dixon and Silver (1981) to guarantee feasibility is a necessary constraint but it is not sufficient for feasibility. Dogramaci et al. (1981) developed a stronger necessary condition for feasibility. For any product the amount of lot size in any period must be greater than or equal to the minimum of the demand of that period and the sum of the excess demands of the subsequent periods. Billington (1986) showed that even with this constraint unidirectional forward pass algorithm may not always generate a feasible schedule and he provided an example where the algorithm proposed by Dogramaci et al. (1981) is unable to generate a feasible schedule. This situation arises usually when the demand is lumpy and exceeds capacity in some periods. Billington (1986) also proposes some modification to overcome this problem.

One of the major drawbacks of the forward pass algorithms is that they first determine the production schedules of earlier periods which may lead to limited possibilities for later periods. Dogramaci et al. (1981) propose a four step heuristic procedure to overcome this complication. In the first step for all $j \in I$ and $t \in \{1, 2, \dots, T\}$ x_{jt} is set to d_{jt} . In the second step, left shifting (shifting the production of some items to an earlier period) and lot batching are done starting with the periods in which highest reduction in cost are possible. Third step ensures feasibility by adjusting the

lots through left shifting operations. Finally in the fourth step the lots are further shifted to the left if cost reduction is possible.

Maes and Wassenhove (1985) compared the heuristic procedures of Lambrecht and Vanderveken (LV), Dixon and Silver (DS), and the four step algorithm of Dogramaci et al. (DPA). They conclude that the difference in performance between the heuristic procedures over a large set of problems is small. As capacity becomes tight and average time between orders increases, however, DPA outperforms DP and LV. On the other hand, when the capacity absorption rates of the products vary DPA does not perform well. In the case of trend in demand DS dominates the others. Although DPA seems to perform better than the others in general, it requires more computational effort than the other heuristic procedures.

Karni and Roll (1982) developed a three phase heuristic procedure. In phase one regardless of capacity consideration a W-W solution is obtained for each item. Infeasibilities are removed by applying adequate backward shift operations to infeasible periods. In phase two cost reduction possibilities are considered while maintaining feasibility. Reduction in the cost can be achieved by eliminating some of the setups. Some of the backward shift operations are generated for this purpose. Reduction in the cost can also be attained as a result of savings in inventory carrying. Some forward shift operations (shifting the lots of some items to a later period)

that may result such reduction are tested. In phase three of the algorithm, structural changes in both directions are considered. Shift operations are similar those generated by Dixon and Silver (1981).

Van Nunen and Wessel (1979) developed another heuristic procedure for a slightly different form of MICDLZ. In their model demand for each item after period T is constant, rather than zero. Their procedure is similar to the procedure developed by Newson (1975). First, for each item they apply a modification of W-W algorithm that handles constant demand after period T. Then infeasibility in any period is eliminated either by shifting the lots to an earlier period and/or to a later period or by breaking the lots and joining them to the lots of earlier or later periods. The following costs are taken into consideration when a lot is to be shifted or split:

- i. cost of extra capacity;
- ii. additional setup cost in case of breaking a lot; and
- iii. additional holding cost when lots are shifted to an earlier period.

2.3.2. Mathematical Programming Based Heuristic Procedures

Thizy and Wassenhove (1985) considered a Lagrangean Relaxation of MICDLZ. In their formulation they relax the capacity constraints and use subgradient optimization to

update the Lagrangean multipliers. When capacity constraints are relaxed the resulting problems are single item uncapacitated problems which can be solved using the W-W algorithm. An upper bound for the problem can be obtained as follows. Assume that production periods for each item are known. MICDLZ then becomes a transportation problem. Furthermore, if the solution to this transportation problem does not violate the capacity constraints than this solution is an upper bound to MICDLZ. Thizy and Wassenhove (1985) reported some computational results. Although their approach outperforms the other heuristic procedures in terms of obtaining a better upper bound, it requires considerably more computational effort than the other heuristic procedures.

Barany, Van Roy and Wolsey (1983) developed a different formulation for (IP_1) . First they consider a single item uncapacitated problem and reformulate this problem by eliminating inventory variables. Then they develop the facets of the convex hull of the feasible region of this reformulated problem. Therefore, the feasible region of (IP_1) can be redefined as the intersection of the convex hull of r single item uncapacitated problems and the capacity constraints. Let this problem be denoted by (FP_1) . (FP_1) is 0-1 mixed integer program. Barany et al. (1983) show that LP relaxation of (FP_1) which is obtained by relaxing integrality requirements gives a stronger lower bound than other relaxations.

Eppen and Martin (1987) reformulated (IP_1) by using the idea of variable redefinition. They first formulate the single item capacitated lot sizing problem using variable redefinition. When single item problem is formulated as such it can be solved by a shortest path algorithm. Then they extend their results to MICDLZ. Let (VIP_1) be the problem obtained by redefining variables. They show that (VIP_1) is a 0-1 mixed integer problem and due to its special structure it can be solved by a shortest path algorithm. However, variable redefinition increases both the number of variables and the number of constraints. Therefore, for large problems it can be computationally prohibitive. They develop a heuristic procedure to overcome this complication. In this heuristic procedure they first solve an LP relaxation of (VIP_1) which gives at least $r-T$ integer values. Then they use branch and bound algorithm on the remaining fractional 0-1 variables. Computational results show that this heuristic procedure is comparable with many of the other heuristic procedures.

2.4. Literature Review on Reverse Convex Programming

Reverse Convex Programming (RCP) is a global optimization problem which has a nonlinear objective function, a set of convex constraints and a set of reverse convex constraints. Note that (P_1) is an (RCP) since its objective function is concave, constraints (3)-(5) and (7)-(9) are linear, and

constraint (6) consists of reverse convex constraints. The earliest work on problems with reverse convex constraints date back to 1966. Rosen (1966), Avriel and Williams (1970), and Avriel (1973) introduced these constraints to the mathematical programming literature. However, most of these early works focused on complimentary geometric programs. The works of Hillestad and Jacobsen (1980a, 1980b) and Thuong and Tuy (1984), and Tuy (1987) brought more insight into (RCP). Hillestad and Jacobsen (1980a, 1980b) proved that when the convex constraints form a polytope and there is a single reverse convex constraint then the optimal solution lies at the intersection of the boundaries of the polytope and a reverse convex constraint provided that feasible region is nonempty and reverse convex constraint is binding. Furthermore, they also showed that if the polytope has at least one edge, then an optimal solution is attained at the intersection of one of the edges of the polytope with the boundary of the reverse convex constraint. Thuong and Tuy (1984) give a similar result. The proof of the theorem for the general convex set case was recently given by Tuy (1987). Since a set of reverse convex constraints can be replaced with a single reverse convex constraint with the addition of some variables, this theorem can be applied to a more general (RCP). Based on these results branch and bound algorithms were developed by Ueing (1972), Hillestad and Jacobsen (1980b), Thoai (1981), Thuong and Tuy (1984), Tuy (1987) and

Fulop (1988). In all of these algorithms reverse convex constraints must first be transformed into a single reverse convex constraint. Branch and Bound algorithms for solving (RCP) without any transformation of the reverse convex constraints have recently been developed by Horst and Dien (1988) and Horst (1988a).

Global optimization problems involving reverse convex constraints have recently drawn much attention in the literature. So far all of the studies have failed to develop an algorithm which has a finite convergence. It seems that much more work needs to be done to come up with applicable algorithms. For an excellent discussion of the algorithms involving reverse convex constraints the reader is referred to Horst (1988b).

CHAPTER 3
BRANCH AND BOUND ALGORITHMS FOR SOLVING
MFDLC AND MFDLZ

In this chapter we present a branch and bound algorithm for solving (P_1) . We then modify this algorithm to obtain a branch and bound algorithm for solving (PZ_1) . The latter algorithm is almost the same as the former algorithm except for few modifications. Therefore, we will develop a branch and bound algorithm for solving (P_1) in the following sections and then we state the modifications in section 3.6. We will henceforth use the term branch and bound algorithm or simply the algorithm without specifying the problem type it is developed for.

Branch and bound algorithm uses linear underestimators for ψ and τ_t , $t \in \{1, 2, \dots, T\}$ to obtain linear subproblems. These linear subproblems are solved to obtain lower bounds on the objective function value of (P_1) at the nodes of the branch and bound tree. We will define two linear underestimators for this purpose. One of the linear underestimators uses the idea of linearization of ψ and τ_t . The second linear underestimator uses the concept of convex envelope to underestimate ψ and τ_t .

Preprocessing data has been suggested as a way to help cut down on computational effort (see for example Van Roy and Wolsey, 1987). As will be shown in this chapter, it is possible to preprocess the data of MFDLC and MFDLZ provided that an upper bound on the objective function of (P_1) and (PZ_1) , respectively are available. Preprocessing scheme we developed for MFDLC and MFDLZ are also similar. Therefore we will only state the preprocessing scheme for MFDLC and state the differences in the preprocessing scheme for MFDLZ.

3.1. Description of the Algorithm

Before discussing the algorithm we will introduce some additional notation, which we hope will help the presentation of the algorithm. Let

$$\Omega_1 = \{(x, y) \in \mathbb{R}^6: x_{j1} - y_{j1} = d_{j1}, y_{jt-1} + x_{jt} - y_{jt} = d_{jt}, y_{jT} = 0, y_{jt} \geq 0, t = 1, 2, \dots, T; j = 1, 2, \dots, r\}$$

$$\Omega_2 = \{(x, y) \in \mathbb{R}^6: 0 \leq x_{jt} \leq U_{jt}, t = 1, 2, \dots, T; j = 1, 2, \dots, r\}$$

$$\Omega_3 = \{(x, y) \in \mathbb{R}^{rT}: \tau_t(x) \leq C_t, t = 1, 2, \dots, T\}$$

We can now restate (P_1) as

$$(P_1) \text{ minimize } f(x, y), \text{ subject to } (x, y) \in (\Omega_1 \cap \Omega_2 \cap \Omega_3).$$

To describe the branch and bound algorithm we will first describe the branch and bound tree that it creates. The nodes of the branch and bound tree will be numbered in the order that they are generated. Let N^p , $p = 0, 1, 2, \dots$, denote the nodes of the tree, where N^0 is the initial node. At each node

of the branch and bound tree each variable x_{jt} , $j = 1, 2, \dots, r$; $t = 1, 2, \dots, T$ is either labeled "free" ($x_{jt} \geq 0$) or "positive" ($x_{jt} > 0$) or "zero" ($x_{jt} = 0$). If a variable is labeled "zero" the constraint $x_{jt} = 0$ is explicitly imposed and if it is labeled "positive" the constraint $x_{jt} > 0$ is not explicitly imposed. Subsequently at each stage $v = 1, 2, \dots$, of the algorithm a node is chosen with at least one free variable. A free variable, x_{sq} , called a separation variable is selected and a branching process is used to partition the node into two new nodes N^{2v-1} and N^{2v} . At node N^{2v-1} , x_{sq} is labeled zero ($x_{sq} = 0$) and at node N^{2v} , x_{sq} is labeled positive ($x_{sq} > 0$).

For any node N^p , $p = 0, 1, 2, \dots$, and any $i \in \{1, 2, \dots, m\}$, let F_i^p be the set of pairs (j, t) , $j \in I_i$, and $t = 1, 2, \dots, T$, for which x_{jt} is labeled free, let P_i^p be the set of pairs (j, t) , $j \in I_i$, $t = 1, 2, \dots, T$, for which x_{jt} is labeled positive and let Z_i^p be the set of pairs (j, t) , $j \in I_i$, $t = 1, 2, \dots, T$, for which x_{jt} is labeled zero. Then associated with each node N^p is the set of solutions $x \in \Omega_2$ which lie in $\Omega_2^p = \Omega_{21}^p \times \Omega_{22}^p \times \dots \times \Omega_{2m}^p$. Here for each $i \in \{1, 2, \dots, m\}$ and each $(x^i, y^i) = (x_1^i, x_2^i, \dots, x_T^i, y_1^i, \dots, y_T^i) \in R^{6i}$ $\Omega_{2i}^p = \{(x^i, y^i) \in R^{6i}: 0 \leq x_{jt} \leq U_{jt} \text{ for all } (j, t) \in F_i^p; 0 < x_{jt} \leq U_{jt} \text{ for all } (j, t) \in P_i^p; x_{jt} = 0 \text{ for all } (j, t) \in Z_i^p\}$.

At each newly created node N^p , the algorithm determines a convex function which underestimates ψ and τ_t , $t \in \{1, 2, \dots, T\}$ on Ω_2^p . Two different convex functions are suggested for this purpose. These functions will be discussed in detail in the

sequel. A lower bound LB^p for f over the solutions $(x,y) \in (\Omega_1 \cap \Omega_2^p \cap \Omega_3)$ is then computed. This lower bound is given by the optimum objective value of the linear subproblem which seeks to minimize an underestimator of the objective function $f(x,y)$ over $(\Omega_1 \cap \Omega_2^{p'} \cap \Omega_3)$ where $(\Omega_2^{p'} \cap \Omega_3) \supseteq (\Omega_2^p \cap \Omega_3)$. Here for all $(x,y) \in \Omega_2^p$, Ω_3 is a relaxation of Ω_3 , and $\Omega_2^{p'}$ is identical to Ω_2^p except $U_{jt} \geq x_{jt} > 0$, for all $(j,t) \in P_1$ in Ω_2^p are replaced by $U_{jt} \geq x_{jt} \geq 0$ for all $(j,t) \in P_1$, for each $i \in \{1, 2, \dots, m\}$. Therefore, $\Omega_2^{p'} \supseteq \Omega_2^p$.

Each subproblem is either infeasible i.e., $(\Omega_1 \cap \Omega_2^{p'} \cap \Omega_3) = \emptyset$ or has an optimal solution (x^p, y^p) . Let UB be the value of the best feasible solution obtained so far. Node N^p is fathomed if any one of the following conditions holds

- i. Subproblem is infeasible
- ii. $LB^p \geq UB$

Note that if $(x^p, y^p) \in (\Omega_1 \cap \Omega_2 \cap \Omega_3)$ then (x^p, y^p) is a feasible solution for (P_1) therefore the upper bound may be updated. UB is an upper bound on f^* , and the solution (x^p, y^p) which satisfies $UB = f(x^p, y^p)$ is called the incumbent solution $(x, y)_{UB}$. As the algorithm proceeds, a candidate list of the nodes that can be branched from is maintained. Any node that can not be fathomed is added to the candidate list.

At each stage v , a node is selected from the candidate list and one of its free variables is used to partition the node into two new nodes. For the newly created nodes, the process of obtaining lower bounds, updating the upper bound

(if necessary) and applying the fathoming tests are repeated. The algorithm terminates when the candidate list becomes empty.

At each node of the branch and bound tree a subproblem is solved which is obtained by underestimating $\psi_{it}(x_t^i)$, for all $i \in \{1, 2, \dots, m\}$ and for all $t \in \{1, 2, \dots, T\}$, and by underestimating $\tau_{it}(x_t^i)$, for all $i \in \{1, 2, \dots, m\}$ and for all $t \in \{1, 2, \dots, T\}$. As stated earlier for each $i \in \{1, 2, \dots, m\}$, each $t \in \{1, 2, \dots, T\}$ and for any $x_t^i \in R^i$, $\tau_{it}(x_t^i)$ and $\psi_{it}(x_t^i)$ are concave and lower semicontinuous functions. We suggest two different underestimating functions for these concave functions. The first of these is obtained by linearizing the corresponding concave function. A subproblem which is obtained by using linear underestimators is denoted by (PR1). The second underestimating function is obtained by constructing the convex envelope of the corresponding concave function over a certain domain. The subproblem which is obtained by using convex envelopes is denoted by (PRC1).

3.2. Construction of Lower Bounding Problems

In this section we develop subproblems (PR1^P) and (PRC1^P) at node N^P of the branch and bound algorithm. (PR1^P) uses linear underestimators for the objective function and the capacity constraints. (PRC1^P) uses the convex envelope of the objective function and the capacity constraints. We construct

a linear underestimator and the convex envelope in sections 3.2.1 and 3.2.2, respectively. The lower bounding subproblem is given in section 3.2.3.

3.2.1. Construction of Linear Underestimator

We now discuss how to obtain a relaxation of constraint (6) and an underestimating function for $f(x,y)$ by using linear underestimators. This involves finding an underestimating function μ_t^p for τ_t for each $t \in \{1, 2, \dots, T\}$ at node N^p and finding an underestimating function f^p for f at node N^p , respectively. Since f and τ_t for any $t \in \{1, 2, \dots, T\}$ are similar in nature, we will first develop an underestimating function μ_t^p and then give an underestimating function f^p which can be developed exactly in the same way as μ_t^p .

In order to construct μ_t^p we will first construct μ_{it}^p for each $i \in \{1, 2, \dots, m\}$. For each $i \in \{1, 2, \dots, m\}$ and each $t \in \{1, 2, \dots, T\}$ and for any vector $x_t^i \in R^{r_i}$, the function μ_{it}^p is given by;

$$\mu_{it}^p(x_t^i) = \left\{ \begin{array}{ll} \sum_{(j,t) \in F_i^p} [(e_{jt}S_i + s_j)/U_{jt} + a_j]x_{jt} & \\ + \sum_{(j,t) \in Z_i^p} a_j x_{jt} & \text{if for all } j \in I_i \\ & (j,t) \in P_i^p \\ \\ S_i + \sum_{(j,t) \in P_i^p} s_j & \text{if for all } j \in I_i \\ & \text{there is at least} \\ & \text{one pair } (j,t) \in P_i^p \\ \\ + \sum_{(j,t) \in F_i^p} (s_j / U_{jt} + a_j)x_{jt} & \\ + \sum_{(j,t) \in P_i^p} a_j x_{jt} + \sum_{(j,t) \in Z_i^p} a_j x_{jt} & \end{array} \right.$$

where for each $i \in \{1, 2, \dots, m\}$ and each $t \in \{1, 2, \dots, T\}$, and for all $(j, t) \in F_i^p$ e_{jt} is a constant satisfying $e_{jt} \geq 0$ and $\sum_{(j,t) \in F_i^p} e_{jt} = 1$.

Then for any vector $x \in R^{KT}$ the function μ_t^p is given by

$$\mu_t^p(x) = \sum_{i=1}^m \mu_{it}^p(x_t^i) \quad (19)$$

We note that for any fixed set of labels F_i^p , P_i^p , and Z_i^p μ_{it}^p , therefore μ_t^p is linear. The following result establishes the underestimating property of μ_t^p .

Theorem 1. For any $(x, y) \in \Omega_{p_2}$, $\mu_t^p(x) \leq \tau_t(x)$ for all $t = 1, 2, \dots, T$.

Proof. Let $i \in \{1, 2, \dots, m\}$ and $t \in \{1, 2, \dots, T\}$ and $x' \in \Omega_{p_2}$. To prove the theorem we will show that $\mu_{it}^p(x'^i_t) \leq \tau_{it}(x'^i_t)$. Then the desired result will follow since i and t were chosen to be arbitrary elements of $\{1, 2, \dots, m\}$ and $\{1, 2, \dots, T\}$, respectively and

$$\mu_t^p(x') = \sum_{i=1}^m \mu_{it}^p(x'^i_t) \text{ and } \tau_t(x') = \sum_{i=1}^m \tau_{it}(x'^i_t)$$

For each $j \in I_i$, (j, t) is an element of either F_i^p , or P_i^p or Z_i^p .

There are four cases to consider.

Case 1. For all $j \in I_i$, $(j, t) \in Z_i^p$

Case 2. For all $j \in I_i$, $(j, t) \in F_i^p$

Case 3. For all $j \in I_i$, either $(j, t) \in Z_i^p$ or $(j, t) \in F_i^p$

Case 4. There is at least one $j \in I_i$ such that $(j, t) \in P_i^p$

In case 1, for all $j \in I_i$, $x'_{jt} = 0$. Therefore,

$$\mu_{it}^p(x'^i_t) = \tau_{it}(x'^i_t) = 0.$$

In case 2, either $x'_{jt} = 0$ for all $j \in I_i$, or there is at least one $j \in I_i$ for which $x'_{jt} > 0$. In the former situation

$$\mu_{it}^p(x'^i_t) = \tau_{it}(x'^i_t) = 0. \text{ In the latter situation,}$$

$$\begin{aligned} \mu_{it}^p(x'^i_t) &= \sum_{(j,t) \in F_i^p} [(e_{jt}S_i + S_j)/U_{jt} + a_j] x'_{jt} \\ &= S_i \left[\sum_{(j,t) \in F_i^p} e_{jt}(x'_{jt}/U_{jt}) \right] + \sum_{(j,t) \in F_i^p} (S_j/U_{jt} + a_j) x'_{jt} \end{aligned} \quad (20)$$

and

$$\begin{aligned} \tau_{it}(x'^i_t) &= S_i + \sum_{j \in I_i} V_{jt}(x'_{jt}) \\ &= S_i + \sum_{(j,t) \in F_i^p} V_{jt}(x'_{jt}) \end{aligned} \quad (21)$$

Comparing (20) and (21) we observe that $\mu_{it}^p(x'^i_t) \leq \tau_{it}(x'^i_t)$.

This is true since $e_{jt} \geq 0$ for all $(j, t) \in F_i^p$ and $\sum_{(j,t) \in F_i^p} e_{jt} = 1$;

and $0 \leq x'_{jt} \leq U_{jt}$ for all $(j, t) \in F_i^p$; and $V_{jt}(x'_{jt}) \geq (S_j/U_{jt} + a_j)x'_{jt}$ when $x'_{jt} \in [0, U_{jt}]$.

In case 3, $\mu_{it}^p(x_t^i) \leq \tau_{it}(x_t^i)$. This follows easily from cases 1 and 2.

In case 4, there is at least one $j \in I_i$ such that $x'_{jt} > 0$. Therefore,

$$\mu_{it}^p(x_t^i) = S_i + \sum_{(j,t) \in (P_i^p \cup Z_i^p)} V_{jt}(x'_{jt}) + \sum_{(j,t) \in F_i^p} (s_j/U_{jt} + a_j)x'_{jt} \quad (22)$$

and

$$\begin{aligned} \tau_{it}(x_t^i) &= S_i + \sum_{j \in I_i} V_{jt}(x'_{jt}) \\ &= S_i + \sum_{(j,t) \in (P_i^p \cup Z_i^p)} V_{jt}(x'_{jt}) + \sum_{(j,t) \in F_i^p} V_{jt}(x'_{jt}) \end{aligned} \quad (23)$$

Comparing (22) and (23) we see that

$$\mu_{it}^p(x_t^i) \leq \tau_{it}(x_t^i)$$

This directly follows from the fact that

$$V_{jt}(x'_{jt}) \geq (s_j/U_{jt} + a_j)x'_{jt} \text{ if } x'_{jt} \in [0, U_{jt}].$$

Put together the above four paragraphs yield the desired result. ■

We now give a linear underestimator for ψ_{it} . For each $i \in \{1, 2, \dots, m\}$ and each $t \in \{1, 2, \dots, T\}$ let $\lambda_{it}^p(x_t^i)$ be a linear underestimating function for $\psi_{it}(x_t^i)$. Then for any vector $x_t^i \in R^{r_i}$, the function $\lambda_{it}^p(x_t^i)$ is given by;

$$\lambda^p_{it}(x^i_t) = \left\{ \begin{array}{ll} \sum_{(j,t) \in F^p_i} [(e_{jt}K_i + k_j)/U_{jt} + c_j] x_{jt} \\ \quad + \sum_{(j,t) \in Z^p_i} c_j x_{jt} & \text{if for all } j \in I_i \\ & (j,t) \notin P^p_i \\ \\ K_i + \sum_{(j,t) \in P^p_i} k_j & \text{if for all } j \in I_i \\ & \text{there is at least} \\ & \text{one pair } (j,t) \in P^p_i \\ \\ \sum_{(j,t) \in F^p_i} (k_j/U_{jt} + c_j) x_{jt} \\ \quad + \sum_{(j,t) \in P^p_i} c_j x_{jt} + \sum_{(j,t) \in Z^p_i} c_j x_{jt} \end{array} \right.$$

Then for any vector $x \in R^T$ the function λ^p is given by

$$\lambda^p(x) = \sum_{t=1}^T \sum_{i=1}^m \lambda^p_{it}(x^i_t)$$

Note that for any fixed set of labels F^p_i , P^p_i and Z^p_i λ^p_{it} therefore $\lambda^p(x,y)$ is linear. The following corollary is a consequence of Theorem 1. Thus the proof is omitted.

Corollary 1. For any $(x,y) \in \Omega^p_2$

$$\lambda^p(x) \leq \psi(x)$$

$$\text{where } \psi(x) = \sum_{i=1}^m \sum_{t=1}^T \psi_{it}(x^i_t) \quad (24)$$

Corollary 2 is an immediate consequence of Corollary 1 and the fact that $h_j y_{jt}$ is linear for all $j \in I$, $t \in \{1, 2, \dots, T\}$.

Corollary 2. For any $(x,y) \in \Omega^p_2$

$$f^p(x,y) \leq f(x,y) \text{ where}$$

$$f^p(x,y) = \lambda^p(x) + \sum_{j=1}^r \sum_{t=1}^T h_j y_{jt}$$

3.2.2. Construction of the Convex Envelope

In this section we show how to develop a relaxation for constraint (6) by constructing the convex envelope for $\tau_t(x)$, $t \in \{1, 2, \dots, T\}$ over appropriate domains. We will also develop an underestimator for the objective function by developing the convex envelope for $\psi(x)$ over appropriate domains.

There are different identical definitions for the convex envelope of a function. The following definition is based on the one given by Falk and Hoffman (1976).

Definition 1. The convex envelope of a function g taken over a nonempty subset D of its domain is that function g^D such that,

- a. g^D is a convex function defined over the convex hull of D ,
- b. $g^D(x) \leq g(x)$ for all $x \in D$,
- c. If g' is a convex function defined over the convex hull of D which satisfies $g'(x) \leq g(x)$ for all $x \in D$, then $g'(x) \leq g^D(x)$ for any x in the convex hull of D .

The convex envelope of a function g over a nonempty subset D of its domain is the pointwise supremum of all convex underestimating functions of g . Therefore, a convex envelope of a function g over a subset D of its domain is the most accurate approximating convex function for g over D among all other possible convex underestimating functions over D .

For each $i \in \{1, 2, \dots, m\}$, for each $t \in \{1, 2, \dots, T\}$ and each $x_t^i \in R^{r_i}$, and at any node N^p we define the following sets

$$F_{it}^p = \{j \in I_i: x_{jt} \text{ is labeled free}\}$$

$$P_{it}^p = \{j \in I_i: x_{jt} \text{ is labeled positive}\}$$

$$Z_{it}^p = \{j \in I_i: x_{jt} \text{ is labeled zero}\}$$

$$\Omega_{2it} = \{x_{it}^i \in R^{r_i}: \text{for all } j \in I_i \ 0 \leq x_{jt} \leq U_{jt}\}$$

$$\Omega_{2it}^p = \{x_{it}^i \in R^{r_i}: 0 \leq x_{jt} \leq U_{jt} \text{ for all } j \in F_{it}^p; x_{jt} = 0 \text{ for all } j \in Z_{it}^p; 0 < x_{jt} \leq U_{jt} \text{ for all } j \in P_{it}^p\}.$$

To construct the convex envelope of τ_t we will first construct the convex envelope of $\tau_{it}(x_{it}^i)$, $i \in \{1, 2, \dots, m\}$ and $t \in \{1, 2, \dots, T\}$ over Ω_{2it} . We will construct the convex envelope of $\psi_{it}(x_{it}^i)$ for all $i \in \{1, 2, \dots, m\}$ and for all $t \in \{1, 2, \dots, T\}$ over Ω_{2it} , which will then be used to construct the convex envelope of $\psi(x)$. Since the structure of $\psi_{it}(x_{it}^i)$ is similar to $\tau_{it}(x_{it}^i)$ for all $i \in \{1, 2, \dots, m\}$ and for all $t \in \{1, 2, \dots, T\}$, we will only give the convex envelopes of $\psi_{it}(x_{it}^i)$ and $\psi(x)$.

To explain the procedure for constructing the convex envelope $\delta_{it}^{\Omega_{2it}}(x_{it}^i)$ of $\tau_{it}(x_{it}^i)$, $i \in \{1, 2, \dots, m\}$ and $t \in \{1, 2, \dots, T\}$ over Ω_{2it} , we need to define r_i -dimensional simplex and simplicial cover of Ω_{2it} . The latter definition was first given by Benson and Erenguc (1988).

Definition 2. Let v^0, v^1, \dots, v^{r_i} be (r_i+1) affinely independent points in R^{r_i} . The convex hull of $\{v^0, v^1, \dots, v^{r_i}\}$ is called a simplex or an r_i -dimensional simplex, and the points v^0, v^1, \dots, v^{r_i} are called vertices of the simplex.

Definition 3. Let $D \subseteq R^i$. A set of r_i -dimensional simplices $\{d_1, d_2, \dots, d_v\}$ is called a simplicial cover of D when

$$D = \bigcup_{j=1}^n d_j.$$

Let $\pi = \{\pi(1), \pi(2), \dots, \pi(r_i)\}$ be any permutation of the integers in r_i . For each $j \in I_i$ let u^j be the unit vector with the integer one in the j^{th} entry and zeros elsewhere. To construct $\delta_{it}^{\Omega_2}$, for any $i \in \{1, 2, \dots, m\}$, and any $t \in \{1, 2, \dots, T\}$, we will find a simplicial cover of Ω_{2it} containing $r_i!$ simplices. Consider the convex hull of the points v^0, v^1, \dots, v^{r_i} , where

$$v^0 = 0 \quad (25)$$

and for each $j \in I_i$

$$v^j = v^{j-1} + U_{\pi(j)t} u^{\pi(j)} \quad (26)$$

The following theorem which defines the simplicial cover of Ω_{2it} is taken from Benson and Erenguc (1988). Hence it is stated without proof.

Theorem 2. Let $N = \{1, 2, \dots, r_i!\}$. The set $\{d_j: j \in N\}$ of all possible $r_i!$ simplices each of whose vertices are given by (25) and (26) for some permutation π of I_i forms a simplicial cover of Ω_{2it} .

For any $i \in \{1, 2, \dots, m\}$ and any $t \in \{1, 2, \dots, T\}$ to construct $\delta_{it}^{\Omega_2}$ for each of the $r_i!$ possible permutations π of r_i , r_i -dimensional simplex whose vertices are given by (25) and (26) is constructed. As stated in Theorem 2, these $r_i!$ simplices form a simplicial cover of Ω_{2it} . For each such simplex d_j , the

unique hyperplane H_{it}^j , which agrees with τ_{it} at each of the vertices of the simplex, is constructed. For each $x_t^i \in R^i$ this hyperplane is given by

$$H_{it}^j(x_t^i) = [(S_i + s_{\pi(1)})/U_{\pi(1)t} + a_{\pi(1)}]x_{\pi(1)t} + \sum_{j=2}^{r_i} [s_{\pi(j)}/U_{\pi(j)t} + a_{\pi(j)}]x_{\pi(j)t} \quad (27)$$

Then for any $x_t^i \in R^i$, the convex envelope of τ_{it} over Ω_{2it} is given by

$$\delta_{it}^{n2}(x_t^i) = \max_{j \in N} \{H_{it}^j(x_t^i)\} \quad (28)$$

where $N = \{1, 2, \dots, r_i!\}$.

The following theorem, which is given by Benson and Erenguc (1988), gives the convex envelope of τ_{it} for each $i \in \{1, 2, \dots, m\}$ and each $t \in \{1, 2, \dots, T\}$. Since properties 2b, 2d, 3b and 4b are equivalent to the properties 2.1a, 2.1b, 2.2 and 2.3, respectively, given in Benson and Erenguc (1988), the proof of the theorem is omitted.

Theorem 3. The convex envelope δ_{it}^{n2} of τ_{it} for each $i \in \{1, 2, \dots, m\}$ and each $t \in \{1, 2, \dots, T\}$ over Ω_{2it} is given, for any $x_t^i \in \Omega_{2it}$, by (28).

Due to theorem 3 it is possible to use the convex envelope results of Benson and Erenguc (1988) to develop the convex envelope of τ_{it} by using $r_i!$ hyperplanes. In many cases, however, it may not be practical to construct $r_i!$ hyperplanes. Due to the special structure of τ_{it} $i \in \{1, 2, \dots, m\}$, and $t \in \{1, 2, \dots, T\}$ we can construct δ_{it}^{n2} by using r_i hyperplanes rather than $r_i!$ hyperplanes. It is

therefore, practically possible to use the convex envelope approach. The following definition is given for this purpose.

Definition 4. Let $\pi_j = \{\pi(1), \pi(2), \dots, \pi(r_1)\}_j$ be the j^{th} set of integers such that for any $k = 1, 2, \dots, r_1$, $1 \leq \pi(k) \leq r_1$, $\pi(1)=j$ and for any $k, l = 1, 2, \dots, r_1$, $\pi(k) \neq \pi(l)$.

Note that there are r_1 such sets. Also note that for any $j \in \{1, 2, \dots, r_1\}$, $\{\pi(1), \pi(2), \dots, \pi(r_1)\}_j$ is an element of the set of all permutations of r_1 . $\pi_1, \pi_2, \dots, \pi_{r_1}$ will, henceforth, be referred to r_1 **subpermutations**. To construct $\delta_{it}^{\Omega_2}$, for each of the r_1 subpermutations which are described above, the r_1 -dimensional simplex whose vertices are given by (25) and (26) are constructed. For each such simplex d_j , $j = 1, 2, \dots, r_1$, the unique hyperplane H_{it}^j , $i \in \{1, 2, \dots, m\}$, $t \in \{1, 2, \dots, T\}$ which agrees with τ_{it} at each of the vertices of the simplex d_j is constructed. This hyperplane for each $x_t^i \in \mathbb{R}^{r_1}$, $i \in \{1, 2, \dots, m\}$ and $t \in \{1, 2, \dots, T\}$ is given by (27), where for any $j \in \{1, 2, \dots, r_1\}$ $\{\pi(1), \pi(2), \dots, \pi(r_1)\}_j$ is given as in definition 4. Then for any $i \in \{1, 2, \dots, m\}$, $t \in \{1, 2, \dots, T\}$ and $x_t^i \in \Omega_{2it}$ the convex envelope of τ_{it} over Ω_{2it} is given by

$$\delta_{it}^{\Omega_2}(x_t^i) = \max_{j \in SN} \{H_{it}^j(x_t^i)\} \quad (29)$$

where $SN = 1, 2, \dots, r_1$.

The following theorem gives the convex envelope $\delta_{it}^{\Omega_2}$ of τ_{it} .

Theorem 4. For any $i \in \{1, 2, \dots, m\}$, $t \in \{1, 2, \dots, T\}$ and $x_t^i \in \Omega_{2it}$ the convex envelope of τ_{it} over Ω_{2it} is given by (29).

Proof. Note that Theorem 4 is similar to Theorem 3 which was proven by Benson and Erenguc (1988). They, however, used $r_i!$ hyperplanes to construct the convex envelope. Therefore, to prove theorem 4 we will show that (28), and (29) are equivalent.

Let $\{\pi'(1), \pi'(2), \dots, \pi'(r_i)\}_k$ and $\{\pi(1), \pi(2), \dots, \pi(r_i)\}_l$ be the k^{th} and l^{th} permutations of r_i , respectively, such that $\pi'(1) = \pi(1)$ and there is at least one k such that $\pi'(k) \neq \pi(k)$. We will show that for all $i \in \{1, 2, \dots, m\}$ and all $t \in \{1, 2, \dots, T\}$ $H_{it}^k(x_t^i) = H_{it}^l(x_t^i)$. For any $i \in \{1, 2, \dots, m\}$ and any $t \in \{1, 2, \dots, T\}$ H_{it}^k and H_{it}^l are given by

$$H_{it}^k(x_t^i) = ((S_i + s_{\pi'(1)})/U_{\pi'(1)t} + a_{\pi'(1)})x_{\pi'(1)t} + \sum_{j=2}^{r_i} (s_{\pi'(j)}/U_{\pi'(j)t} + a_{\pi'(j)})x_{\pi'(j)t} \quad (30)$$

$$H_{it}^l(x_t^i) = ((S_i + s_{\pi(1)})/U_{\pi(1)t} + a_{\pi(1)})x_{\pi(1)t} + \sum_{j=2}^{r_i} (s_{\pi(j)}/U_{\pi(j)t} + a_{\pi(j)})x_{\pi(j)t} \quad (31)$$

Since it is assumed that $\pi'(1) = \pi(1)$,

$$((S_i + s_{\pi'(1)})/U_{\pi'(1)t} + a_{\pi'(1)})x_{\pi'(1)t} = ((S_i + s_{\pi(1)})/U_{\pi(1)t} + a_{\pi(1)})x_{\pi(1)t} \quad (32)$$

By definition 4, for any $j \in \{1, 2, \dots, r_i\}$ there exists some $k \in \{1, 2, \dots, r_i\}$ such that $\pi'(j) = \pi(k)$. Therefore,

$$(s_{\pi'(j)}/U_{\pi'(j)t} + a_{\pi'(j)})x_{\pi'(j)t} = (s_{\pi(k)}/U_{\pi(k)t} + a_{\pi(k)})x_{\pi(k)t} \quad (33)$$

Also by definition 4, for any $m, n \in \{1, 2, \dots, r_i\}$, $\pi'(m) \neq \pi'(n)$ and $\pi(m) \neq \pi(n)$. This together with (33) implies that

$$\sum_{j=2}^{r_i} (s_{\pi'(j)}/U_{\pi'(j)t} + a_{\pi'(j)})x_{\pi'(j)t} = \sum_{j=2}^{r_i} (s_{\pi(j)}/U_{\pi(j)t} + a_{\pi(j)})x_{\pi(j)t} \quad (34)$$

Therefore, (32) and (34) imply that (28) and (29) are equivalent. Since k and l are arbitrarily chosen permutations of r_i provided that $\pi'(1)=\pi(1)$, it is shown that for any given $\pi(1)$ the same hyperplane is obtained for any permutation of $\{\pi(2), \pi(3), \dots, \pi(r_i)\}$. Therefore, $r_i!$ hyperplanes reduce to r_i hyperplanes, thus the proof is complete. ■

We now explain the procedure for constructing the convex envelope $\delta^{n2p}_{it}(x^i_t)$ of $r_{it}(x^i_t)$, for each $i \in \{1, 2, \dots, m\}$ and $t \in \{1, 2, \dots, T\}$ at node N^p over Ω^p_{2it} of the branch and bound tree. There are five cases to consider. These are:

- (i) $P^p_{it} = \emptyset$ and $Z^p_{it} = \emptyset$,
- (ii) $P^p_{it} = \emptyset$, $Z^p_{it} \neq \emptyset$ and $F^p_{it} \neq \emptyset$,
- (iii) $P^p_{it} = \emptyset$, $F^p_{it} = \emptyset$ and $Z^p_{it} \neq \emptyset$,
- (iv) $F^p_{it} = \emptyset$ and $P^p_{it} \neq \emptyset$,
- (v) $F^p_{it} \neq \emptyset$ and $P^p_{it} \neq \emptyset$.

Case (i). This is the case which is used to construct the convex envelope defined by (28). Hence Theorem 4 is used to construct the convex envelope.

Case (ii). Since $P^p_{it} = \emptyset$, $F^p_{it} \neq \emptyset$ and $Z^p_{it} \neq \emptyset$, $\Omega^p_{2it} = \{x^i_t \in R^i: x_{jt} = 0 \text{ for all } j \in Z^p_{it}; 0 \leq x_{jt} \leq U_{jt} \text{ for all } j \in F^p_{it}\}$. Define a function q^p_{it} over Ω^p_{2it} as

$$q^p_{it}(x^i_t) = G_{it}(x^i_t) + \sum_{j \in F^p_{it}} V_{jt}(x_{jt}) \text{ for each } x^i_t \in \Omega^p_{2it} \quad (35)$$

Then for each $x^i_t \in \Omega^p_{2it}$

$$\delta^{n2p}_{it}(x^i_t) = q^{n2p}_{it}(x^i_t) \quad (36)$$

where $q^{n2p}_{it}(x^i_t)$ is the convex envelope of $q^p_{it}(x^i_t)$ over Ω^p_{2it} .

Notice that q^{n2p}_{it} can be constructed by the procedure used to construct (28). This follows from Theorem 4 and the facts that for each $i \in \{1, 2, \dots, m\}$ and each $t \in \{1, 2, \dots, T\}$ q_{it} and τ_{it} are of the same form and Ω^p_{2it} and Ω_{2it} are of the same form.

Case (iii). In this case for each $x^i_t \in \Omega^p_{2it}$, $\delta^{n2p}_{it}(x^i_t) = 0$.

Case (iv). For each $x^i_t \in \Omega^p_{2it}$,

$$\delta^{n2p}_{it}(x^i_t) = S_i + \sum_{j \in P^p_{it}} s_j + a_j x_{jt} \quad (37)$$

Case (v). In this case for each $x^i_t \in \Omega^p_{2it}$,

$$\delta^{n2p}_{it}(x^i_t) = S_i + \sum_{j \in P^p_{it}} V_{jt}(x_{jt}) + \sum_{j \in F^p_{it}} (s_j/U_{jt} + a_j) x_{jt} \quad (38)$$

To show that the procedures given in cases (ii)-(v) are the correct formulas for δ^{n2p}_{it} , we give the following theorem.

Theorem 5. For any $i \in \{1, 2, \dots, m\}$, $t \in \{1, 2, \dots, T\}$ and for any $x^i_t \in \Omega^p_{2it}$ exactly one of the following statement is true

- (i) If $P^p_{it} = \emptyset$, $Z^p_{it} \neq \emptyset$ and $F^p_{it} \neq \emptyset$ then the convex envelope δ^{n2p}_{it} of τ_{it} over Ω^p_{2it} is given by (36)
- (ii) If $P^p_{it} = \emptyset$, $Z^p_{it} \neq \emptyset$ and $F^p_{it} = \emptyset$ then $\delta^{n2p}_{it}(x^i_t) = 0$
- (iii) If $P^p_{it} \neq \emptyset$ and $F^p_{it} = \emptyset$ then the convex envelope δ^{n2p}_{it} of τ_{it} over Ω^p_{2it} is given by (37)
- (iv) If $P^p_{it} \neq \emptyset$, $F^p_{it} \neq \emptyset$ then the convex envelope δ^{n2p}_{it} of τ_{it} over Ω^p_{2it} is given by (38).

Proof. Let $x_t^i \in \Omega_{2it}^p$ for any $i \in \{1, 2, \dots, m\}$ and any $t \in \{1, 2, \dots, T\}$. Since i and t are arbitrarily chosen elements of $\{1, 2, \dots, m\}$ and $\{1, 2, \dots, T\}$, respectively, the proof of all parts will be completed when each part is proven for any $i \in \{1, 2, \dots, m\}$ and for any $t \in \{1, 2, \dots, T\}$.

(i). Since $P_{it}^p = \emptyset$, and $F_{it}^p \neq \emptyset$, $\Omega_{2it}^p = \{x_t^i \in R^{F_i}: x_{jt} = 0 \text{ for all } j \in Z_{it}^p; 0 \leq x_{jt} \leq U_{jt} \text{ for all } j \in F_{it}^p\}$. Therefore for any $x_t^i \in \Omega_{2it}^p$, $\tau_{it}(x_t^i)$ is given by

$$\tau_{it}(x_t^i) = G_{it}(x_t^i) + \sum_{j \in F_{it}^p} V_{jt}(x_{jt}) \quad (39)$$

Since (35) and (39) are equivalent the convex envelope $\delta^{\Omega_{2it}^p}$ of τ_{it} over Ω_{2it}^p is given by (36).

(ii). Since $P_{it}^p = \emptyset$, $F_{it}^p = \emptyset$ and $Z_{it}^p \neq \emptyset$ then $x_{jt} = 0$ for all $j \in I_i$. Therefore, from the definition of Ω_{2it}^p , if $x_t^i \in \Omega_{2it}^p$ then $x_t^i = 0$. This implies that $\tau_{it}(x_t^i) = 0$ for each $x_t^i \in \Omega_{2it}^p$. Therefore, τ_{it}^p is a constant over Ω_{2it}^p . This implies, from definition 1, that the convex envelope $\delta^{\Omega_{2it}^p}$ of τ_{it} over Ω_{2it}^p is τ_{it} itself. This completes the proof of (ii).

(iii). Since $P_{it}^p \neq \emptyset$ and $F_{it}^p = \emptyset$, if $x_t^i \in \Omega_{2it}^p$ then for each $j \in I_i$ either $x_{jt} = 0$ or $x_{jt} > 0$. This implies that for each $x_t^i \in \Omega_{2it}^p$, τ_{it} is given by

$$\tau_{it}(x_t^i) = S_i + \sum_{j \in P_{it}^p} S_j + a_j x_{jt} \quad (40)$$

Therefore, for fixed P_{it}^p , F_{it}^p and $x_t^i \in \Omega_{2it}^p$ τ_{it} is a linear function. Since the convex envelope of a linear function is itself, the convex envelope $\delta^{\Omega_{2it}^p}$ of τ_{it} over Ω_{2it}^p is given by (37).

(iv). Since $P_{it}^p \neq \emptyset$ there is at least one $x_{jt} > 0$, $j \in I_1$.

Then,

$$\tau_{it}(x_{jt}^i) = S_i + \sum_{j \in P_{it}^p}^r (s_j + a_j x_{jt}) + \sum_{j \in F_{it}^p} V_{jt}(x_{jt}) \quad (41)$$

It is a well known result [Falk and Soland, 1969] that for any $j \in I$ and $t \in \{1, 2, \dots, T\}$ the convex envelope of $V_{jt}(x_{jt})$ over $[0, U_{jt}]$ is given by $(s_j/U_{jt} + a_j)x_{jt}$. For fixed F_{it}^p and P_{it}^p

$\sum_{j \in P_{it}^p}^r (s_j + a_j x_{jt})$ is a linear function and the convex envelope of

a linear function is itself. Furthermore, the convex envelope of the sum of two functions is the sum of the convex envelopes of these two functions (see theorem 6 below). Therefore the convex envelope of τ_{it} over $\cap \Omega_{2it}^p$ is given by

$$\delta_{it}^{n2p}(x_{jt}^i) = S_i + \sum_{j \in P_{it}^p} (s_j + a_j)x_{jt} + \sum_{j \in F_{it}^p} (s_j/U_{jt} + a_j)x_{jt} \quad (42)$$

This completes the proof since (42) and (38) are equivalent. ■

For any $i \in \{1, 2, \dots, m\}$, and any $t \in \{1, 2, \dots, T\}$ let NF be the cardinality of the set F_{it}^p . In other words NF denotes the number of products in family i in period t which has not yet been labeled. To summarize, at any node N^p the convex envelope δ_{it}^{n2p} of τ_{it} $i \in \{1, 2, \dots, m\}$ and $t \in \{1, 2, \dots, T\}$ over Ω_{2it}^p is given by

$$\delta^{n2p}_{it}(x^i_t) = \begin{cases} \max_{j \in \{1, 2, \dots, NF\}} \{H^j_{it}(x^i_t)\} & \text{if } P^p_i = \emptyset \text{ and } F^p_i \neq \emptyset \\ S_i + \sum_{j \in P^p_{it}} (s_j + a_j x_{jt}) & \\ + \sum_{j \in F^p_{it}} (s_j/U_{jt} + a_j) x_{jt} & \text{if } P^p_i \neq \emptyset \text{ and } F^p_i \neq \emptyset \quad (43) \\ S_i + \sum_{j \in P^p_i} (s_j + a_j x_{jt}) & \text{if } P^p_i \neq \emptyset \text{ and } F^p_i = \emptyset \\ 0 & \text{otherwise} \end{cases}$$

where H^j_{it} is given by (27).

We will use the following theorem, which is given in Falk and Soland (1969), to give the convex envelope of τ_t .

Theorem 6. Let $f(x) = \sum_{i=1}^n f_i(x^i)$ be a real-valued function.

Suppose that g_i is the convex envelope of f_i taken on T_i for each $i=1, 2, \dots, n$. Let $T=T_1 \times T_2 \times \dots \times T_n$. For any $x = (x^1, x^2, \dots, x^n) \in T$, where $x^i \in T_i$ for each $i=1, 2, \dots, n$, define a

function g by $g(x) = \sum_{i=1}^n g_i(x^i)$. Then $g(x)$ is the convex

envelope of $f(x)$ taken over T .

We are now in a position to give the convex envelope of $\tau_t(x)$ as a corollary, which is a direct consequence of theorems 5 and 6, and the fact that convex envelope of a function over a certain domain underestimates that function over that domain.

Corollary 3. For any $t \in \{1, 2, \dots, T\}$ and $x \in \Omega^p_2$ let $\Delta^p_t(x)$ be the convex envelope of $\tau_t(x, y)$ at node N^p . Then for any $t \in \{1, 2, \dots, T\}$,

$$\Delta^p_t(x) = \sum_{i=1}^m \delta^{n2p}_{it}(x^i_t) \leq \tau_t(x) \quad (44)$$

We now give the convex envelope for the objective function of MFDLC over Ω_2 . To construct the convex envelope of ψ over Ω_2 , we need to construct the convex envelope of $\psi_{it}(x^i_t)$ for any $i \in \{1, 2, \dots, m\}$, any $t \in \{1, 2, \dots, T\}$ and for any $x^i_t \in \Omega_{2it}$. Since for each $i \in \{1, 2, \dots, m\}$ and $t \in \{1, 2, \dots, T\}$ $\psi_{it}(x^i_t)$ and $\tau_{it}(x^i_t)$ are similar in nature and the underestimating function is constructed over the same hyperrectangle, Ω_{2it} , the convex envelope of ψ_{it} can be constructed the same way as the convex envelope of τ_{it} . But, first we need to show that theorem 3 holds for ψ_{it} . This is necessary because property 2c is slightly different from property 2.1b of Benson and Erenguc (1988), therefore the proof of theorem 3 for ψ_{it} does not directly follow from the proof of theorem 3.1 given in Benson and Erenguc (1988). Note that in property 2c $\psi_{it}(A^i_t) \geq 0$ for any $A^i_t \neq \emptyset$. Theorem 3, however, requires that $\psi_{it}(A^i_t) > 0$ for any $A^i_t \neq \emptyset$. But, we note that if $K^i = 0$ then $\psi_{it}(x^i_t) = \sum_{j \in I_i} M_{jt}(x_{jt})$. Therefore, ψ_{it} is a concave separable function and the convex envelope of ψ_{it} can be obtained as explained in the proof of theorem 5 part (iv). Due to the similarities of τ_{it} and ψ_{it} , $i \in \{1, 2, \dots, m\}$, $t \in \{1, 2, \dots, T\}$, we can give the convex envelope of ψ_{it} at node N^p without proof. Let $\rho^{n2p}_{it}(x^i_t)$, $i \in \{1, 2, \dots, m\}$, $t \in \{1, 2, \dots, T\}$ and $x^i_t \in \Omega^p_{2it}$ be the convex envelope of $\psi_{it}(x^i_t)$ over Ω^p_{2i} at node

N^P . Then,

$$\rho^{n2p}_{it}(x^i_t) = \begin{cases} \max_{j \in \{1, 2, \dots, NF\}} \{O^j_{it}(x^i_t)\} & \text{if } P^P_{it} = \emptyset \text{ and } F^P_{it} \neq \emptyset \\ S_i + \sum_{j \in P^P_{it}} (k_j + a_j x_{jt}) \\ \quad + \sum_{j \in F^P_{it}} (S_j / U_{jt} + c_j) x_{jt} & \text{if } P^P_{it} \neq \emptyset \text{ and } F^P_{it} \neq \emptyset \\ S_i + \sum_{j \in P^P_{it}} (k_j + a_j x_{jt}) & \text{if } P^P_{it} \neq \emptyset \text{ and } F^P_{it} = \emptyset \end{cases} \quad (45)$$

where for any $j \in \{1, 2, \dots, SN\}$, $i \in \{1, 2, \dots, m\}$ and $t \in \{1, 2, \dots, T\}$

$$O^j_{it}(x^i_t) = [(S_i + S_{\pi(1)}) / U_{\pi(1)t} + a_{\pi(1)}] x_{\pi(1)t} + \sum_{j=2}^{r_1} [S_{\pi(j)} / U_{\pi(j)t} + a_{\pi(j)}] x_{\pi(j)t}$$

As a direct consequence of theorems 5 and 6, and the fact that for any $j \in I$ $h_j y_j$ is a linear function we have the following corollary

Corollary 4. For any $(x, y) \in N^P_2$ let $v^P(x, y)$ be the convex envelope of $f(x, y)$ over N^P_2 at node N^P . Then,

$$v^P(x, y) = \sum_{t=1}^T \sum_{i=1}^m \rho^{n2p}_{it}(x^i_t) + \sum_{t=1}^T \sum_{j=1}^r h_j y_{jt} \leq f(x, y) \quad (46)$$

3.2.3. Lower Bounding Subproblems

As we mentioned earlier, a lower bound LB^P is calculated for f over the solutions $(x, y) \in (\Omega_1 \cap N^P_2 \cap \Omega_3)$ at node N^P . This lower bound can be obtained either by solving $(PR1^P)$ or by solving $(PRC1^P)$ at node N^P . Note that $(PR1^P)$ and $(PRC1^P)$ differs only in the underestimating functions they used for the objective function and the capacity constraints. Let

$\Gamma_{it}^p(x_t^i)$ $i \in \{1, 2, \dots, m\}$ and $t \in \{1, 2, \dots, T\}$ be an underestimator for $r_{it}(x_t^i)$ at node N^p defined by either (19) or (44). Let $\sigma_{it}^p(x_t^i)$ $i \in \{1, 2, \dots, m\}$ and $t \in \{1, 2, \dots, T\}$ be an underestimator for $\psi_{it}(x_t^i)$ at node N^p defined by either (24) or (45). Let

$$\sigma^p(x, y) = \sum_{t=1}^T \sum_{i=1}^m \sigma_{it}^p(x_t^i) + \sum_{t=1}^T \sum_{j=1}^r h_{jt} y_{jt} \text{ and } \Gamma_t^p(x) = \sum_{i=1}^m \Gamma_{it}^p(x_t^i).$$

Now let us define

$$\Omega_3^p = \{(x, y) \in R^B: \Gamma_t^p(x) \leq C_t, t=1, 2, \dots, T\}.$$

The following corollary is an immediate consequence of theorem 1 and corollary 3 and the fact that $\Omega_2^{p'} \supseteq \Omega_2^p$.

Corollary 5. $(\Omega_2^{p'} \cap \Omega_3^p) \supseteq (\Omega_2^p \cap \Omega_3^p) \supseteq (\Omega_2^p \cap \Omega_3)$

Let $(PL1^p)$ be a subproblem at node N^p . Note that $(PL1^p)$ is either $(PR1^p)$ or $(PRC1^p)$. We now establish the lower bounding property of $(PL1^p)$ at node N^p . Let $v(\cdot)$ be the optimal objective function value of problem (\cdot) . We now define the following problems.

- (P1) $\min f(x, y)$
s.t. $(x, y) \in (\Omega_1 \cap \Omega_2^p \cap \Omega_3)$
- (P2) $\min \sigma(x, y)$
s.t. $(x, y) \in (\Omega_1 \cap \Omega_2^p \cap \Omega_3)$
- (P3) $\min \sigma(x, y)$
s.t. $(x, y) \in (\Omega_1 \cap \Omega_2^{p'} \cap \Omega_3^p)$

We note that by corollary 2 and corollary 4 and the fact that $(\Omega_1 \cap \Omega_2^p \cap \Omega_3) \subseteq \Omega_2^p$ $v(P2) \leq v(P1)$. By corollary 5, $v(P3) \leq v(P2)$ which implies that $v(PL1^p) = v(P3) \leq v(P2)$. Therefore,

(PL1^P) is a lowerbounding subproblem at node N^P. We are now in a position to define (PR1^P) and (PRC1^P). (PR1^P) is given by

$$\begin{aligned} (\text{PR1}^P) \quad & \min f(x, y) \\ \text{s.t.} \quad & (x, y) \in (\Omega_1 \cap \Omega^{P'_2} \cap \Omega^{P_3}) \end{aligned}$$

where in Ω^{P_3} $\Gamma_t^P(x) = \mu_t^P(x)$ for all $t \in \{1, 2, \dots, T\}$. Since for any fixed set of labels F_{i1}^P , P_{i1}^P and Z_{i1}^P $f(x, y)$ and $\mu_t^P(x)$ are linear, (PR1^P) is a linear program. (PRC1^P) is given by

$$\begin{aligned} (\text{PRC1}^P) \quad & \min v(x, y) \\ \text{s.t.} \quad & (x, y) \in (\Omega_1 \cap \Omega^{P'_2} \cap \Omega^{P_3}) \end{aligned}$$

where in Ω^{P_3} , $\Gamma_t^P(x) = \Delta_t(x)$. Note that for a fixed set of labels F_{it}^P , P_{it}^P and Z_{it}^P , and for each $j \in \{1, 2, \dots, NF\}$ H_{it}^j and O_{it}^j are linear. Therefore, (PRC1^P) may contain the maximization of linear functions. It is a well known fact that the maximum of linear functions is a piecewise linear and convex function. We will now show that (PRC1^P) is in deed a linear program. Without loss of generality we will assume that $P_{it}^P = \emptyset$ and $Z_{it}^P = \emptyset$. (PRC1^P) can be written as follows

$$\begin{aligned} (\text{PRC1}^P) \quad & \min \sum_{i=1}^m \sum_{t=1}^T \max_{j=1, 2, \dots, r_1} \{O_{it}^j\} + \sum_{j=1}^r \sum_{t=1}^T h_j y_{jt} \\ \text{s.t.} \quad & y_{jt-1} + x_{jt} - y_{jt} = d_{jt} & \forall j, t \\ & \sum_{i=1}^m \max_{j=1, 2, \dots, r_1} \{H_{it}^j\} \leq C_t & \forall t \\ & 0 \leq x_{jt} \leq U_{jt} & \forall j, t \\ & y_{jt} \geq 0 & \forall j, t \\ & y_{jT} = 0 & \forall j \end{aligned}$$

(PRC1^P) can futher be written as follows

$$(PRC1^P) \min \sum_{i=1}^m \sum_{t=1}^T t_{it} + \sum_{j=1}^r \sum_{t=1}^T h_j y_{jt}$$

$$\begin{aligned} \text{s.t. } O_{it}^j &\leq t_{it} & \forall t, i \text{ and } j \in I_i \\ Y_{jt-1} + x_{jt} - y_{jt} &= d_{jt} & \forall j, t \\ H_{it}^j &\leq p_{it} & \forall t, i \text{ and } j \in I_i \\ \sum_{i=1}^m p_{it} &\leq C_t & \forall t \\ 0 &\leq x_{jt} \leq U_{jt} & \forall j, t \\ y_{jt} &\geq 0 & \forall j, t \\ y_{jT} &= 0 & \forall j \\ t_{it} &\geq 0 & \forall i, t \\ p_{it} &\geq 0 & \forall i, t \end{aligned}$$

Therefore, $(PRC1^P)$ is also a linear program. Since the convex envelope of a function always gives the most accurate underestimating convex function over a certain domain, the subproblem $(PRC1^P)$ which uses the convex envelopes given by (44) and (46) gives a tighter lower bound than $(PR1^P)$. However, as seen from the formulation, $(PRC1^P)$ has more variables and constraints than $(PR1^P)$. For example, $(PRC1^0)$ has $2mT$ more variables and $T(2r-1)$ more constraints than $(PR1^0)$ at node N^0 . Once an item in family i , $i \in \{1, 2, \dots, m\}$ for period t , $t \in \{1, 2, \dots, T\}$ is fixed at a positive level, $2r_i-1$ of these constraints become redundant. Therefore, they can be eliminated from the linear program.

3.3. The Branching Process

At stage v of the algorithm, a node N^p which satisfies $LB^p < UB$ is removed from the candidate list. The optimal solution (x^p, y^p) of $(PR1^p)$ or $(PRC1^p)$ found earlier is retrieved. The branching process is then implemented to partition node N^p into two new nodes N^{2v-1} and N^{2v} . To generate the two new nodes, a separation variable x_{kq} is determined. Any variable x_{kq} , $k \in I$, $q \in \{1, 2, \dots, T\}$ which satisfies one of the following conditions may be chosen as the separation variable.

- a) Choose any $(k, q) \in F_i^p$, such that $x_{kq}^p > 0$, if for all $j \in I_i$, $(j, q) \notin P_i^p$, and not all pairs $(j, q) \in F_i^p$ have $x_{jq}^p = U_{jq}$
- b) Choose any $(k, q) \in F_i^p$ such that $0 < x_{kq}^p < U_{kq}$ and $s_k > 0$, if for some $j \in I_i$, $(j, q) \in P_i^p$.

The existence of the pair (k, q) is given by the following Lemma.

Lemma 1. Let N^p be a node removed from the candidate list for partitioning and (x^{ip_t}, y^{ip_t}) be the solution to subproblem $(PL1^p)$. Then $\sigma_{it}^p(x^{ip_t}) < \psi_{it}(x^{ip_t})$ for some $i \in \{1, 2, \dots, m\}$ for some $t \in \{1, 2, \dots, T\}$ and $x^{ip_t} \in \Omega_{2it}^p$. Furthermore, for any $i \in \{1, 2, \dots, m\}$ and $t \in \{1, 2, \dots, T\}$ for which $\sigma_{it}^p(x^{ip_t}) < \psi_{it}(x^{ip_t})$ is satisfied there exists a pair $(j, t) \in F_i^p$ which satisfies either rule (a) or rule (b).

Proof. Let $f(x^p, y^p)$ be the objective function value of (P_1) at node N^p . We first note that if node N^p is to be

partitioned, then $LB^p < UB$ and $(\Omega_1 \cap \Omega^{p'}_2 \cap \Omega^p_3) \neq \emptyset$. By definition of LB^p , since $UB \leq f(x^p, y^p)$, this implies that

$$\sum_{t=1}^T \sum_{i=1}^m \sigma^p_{it}(x^{ip}_t) + \sum_{t=1}^T \sum_{j=1}^r h_j y^p_{jt} < \sum_{t=1}^T \sum_{i=1}^m \psi_{it}(x^{ip}_t) + \sum_{t=1}^T \sum_{j=1}^r h_j y^p_{jt}$$

Therefore, $\sigma^p_{it}(x^{ip}_t) < \psi_{it}(x^{ip}_t)$ for some $i \in \{1, 2, \dots, m\}$ and for some $t \in \{1, 2, \dots, T\}$. Let $i \in \{1, 2, \dots, m\}$ and $t \in \{1, 2, \dots, T\}$ satisfy $\sigma^p_{it}(x^{ip}_t) < \psi_{it}(x^{ip}_t)$. There are four possible cases concerning P^p_{it} , Z^p_{it} , F^p_{it} , and $x^{ip}_t \in \Omega^p_{2it}$. These cases are as follows:

Case 1. Either $F^p_{it} = \emptyset$; or for all $j \in F^p_{it}$ $x^p_{jt} = 0$; or for all $j \in F^p_{it}$ $x^p_{jt} = U_{jt}$; or for all $j \in F^p_{it}$ $x^p_{jt} = 0$ or $x^p_{jt} = U_{jt}$ and $P^p_{it} \neq \emptyset$ and for all $j \in P^p_{it}$ $x^p_{jt} > 0$.

Case 2. Either $F^p_{it} = \emptyset$; or for all $j \in F^p_{it}$ $x^p_{jt} = 0$; or for all $j \in F^p_{it}$ $x^p_{jt} = U_{jt}$; or for all $j \in F^p_{it}$ $x^p_{jt} = 0$ or $x^p_{jt} = U_{jt}$ and $P^p_{it} \neq \emptyset$ and for some $j \in P^p_{it}$ $x^p_{jt} = 0$.

Case 3. Either $F^p_{it} = \emptyset$; or for all $j \in F^p_{it}$ $x^p_{jt} = 0$; or for all $j \in F^p_{it}$ $x^p_{jt} = U_{jt}$; or for all $j \in F^p_{it}$ $x^p_{jt} = 0$ or $x^p_{jt} = U_{jt}$ and $P^p_{it} \neq \emptyset$ and for all $j \in P^p_{it}$ either $x^p_{jt} = 0$ or $x^p_{jt} = U_{jt}$.

Case 4. Either $P^p_{it} = \emptyset$, or $P^p_{it} \neq \emptyset$ and $F^p_{it} \neq \emptyset$ and for some $j \in F^p_{it}$, $0 < x^p_{jt} < U_{jt}$.

To prove that there exists a $j \in F^p_{it}$ which satisfies rule (a) or rule (b) we will show that cases 1 through 3 cannot occur. When $P^p_{it} = \emptyset$ in Case 4 rule (a) is obtained and when $P^p_{it} \neq \emptyset$ in Case 4 rule (b) is obtained.

Case 1. Let FU_{it}^p be the set of j such that $j \in F_{it}^p$ and $x_{jt}^p = U_{jt}$. Let FZ_{it}^p be the set of j such that $j \in F_{it}^p$ and $x_{jt}^p = 0$. Then,

$$\begin{aligned}\sigma_{it}^p(x_{it}^{ip}) &= K_i + \sum_{j \in P_{it}^p} V_{jt}(x_{jt}^p) + \sum_{j \in FU_{it}^p} (k_j + c_j x_{jt}^p) \\ &= \psi_{it}(x_{it}^{ip})\end{aligned}$$

which contradicts the fact that $\sigma_{it}^p(x_{it}^{ip}) < \psi_{it}(x_{it}^{ip})$. Therefore Case 1 cannot occur.

Case 2. Let $PP_{it}^p = \{j \in P_{it}^p \text{ such that } x_{jt}^p > 0\} \neq \emptyset$ and let $PZ_{it}^p = \{j \in P_{it}^p \text{ such that } x_{jt}^p = 0\} \neq \emptyset$. Then,

$$\sigma_{it}^p(x_{it}^{ip}) = K_i + \sum_{j \in PP_{it}^p} (k_j + c_j x_{jt}^p) + \sum_{j \in FU_{it}^p} (k_j + c_j x_{jt}^p) + \sum_{j \in PZ_{it}^p} (k_j + c_j x_{jt}^p)$$

However, $\psi_{it}(x_{it}^{ip}) = K_i + \sum_{j \in (PP_{it}^p \cup FU_{it}^p)} V_{jt}(x_{jt}^p)$. Therefore, $\sigma_{it}^p(x_{it}^{ip}) \geq \psi_{it}(x_{it}^{ip})$.

Since this contradicts the fact that $\sigma_{it}^p(x_{it}^{ip}) < \psi_{it}(x_{it}^{ip})$, Case 2 cannot occur.

Case 3. Let $PU_{it}^p = \{j \in P_{it}^p \text{ such that } x_{jt}^p = U_{jt}\} \neq \emptyset$. Then

$$\sigma_{it}^p(x_{it}^{ip}) = K_i + \sum_{j \in PU_{it}^p} (k_j + c_j x_{jt}^p) + \sum_{j \in FU_{it}^p} (k_j + c_j x_{jt}^p) + \sum_{j \in PZ_{it}^p} (k_j + c_j x_{jt}^p)$$

However, $\psi_{it}(x_{it}^{ip}) = K_i + \sum_{j \in (PU_{it}^p \cup FU_{it}^p)} V_{jt}(x_{jt}^p)$. Therefore,

$$\sigma_{it}^p(x_{it}^{ip}) \geq \psi_{it}(x_{it}^{ip}).$$

Since this contradicts the fact that $\sigma_{it}^p(x_{it}^{ip}) < \psi_{it}(x_{it}^{ip})$, case 3 cannot occur. ■

3.4. A Formal Statement of the Branch and Bound Algorithm

Step 0. Initialization

- a. $v \leftarrow 0$. Candidate list is empty $UB = \infty$.
 $P_i^0 = Z_i^0 = \emptyset$ for all $i \in \{1, 2, \dots, m\}$
 $F_i^0 = \{(j, t) : j = 1, 2, \dots, r, \text{ and } t = 2, 3, \dots, T\}$.
- b. Solve either $(PR1^0)$ or $(PRC1^0)$ to obtain (x^0, y^0) and LB^0 . If $(x^0, y^0) \in (\Omega_1 \cap \Omega_2 \cap \Omega_3)$, set $UB = f(x^0, y^0)$ and $(x, y)_{UB} = (x^0, y^0)$. Add N^0 to the candidate list.
 $v \leftarrow 1$ and go to step 1.

Step 1. If candidate list is empty go to step 3. Otherwise go to step 2.

Step 2. Remove any node N^p from the candidate list. If $LB^p \geq UB$ go to step 1. Otherwise generate nodes N^{2v-1} and N^{2v} . Obtain LB^{2v-1} and LB^{2v} . Update UB and $(x, y)_{UB}$ if possible. For each $j = 2v-1, 2v$, if $LB^j < UB$ add node N^j to the candidate list. $v \leftarrow v+1$. Go to step 1.

Step 3. Termination. $(x, y)_{UB}$ is an optimal solution for problem (P_1) with a value of $f^* = UB$. Stop.

We make the following remarks about the formal statement of the algorithm.

1) Since backlogging is not permitted in MFDLC, those products with positive demand in period 1 must be produced in period 1. Therefore, these products should be labeled positive in the initialization step.

2) If a feasible solution to (P_1) is available, possibly

from a heuristic procedure, then UB is set equal to the value of this solution, and the solution is stored in $(x, y)_{UB}$ in the initialization step.

3) In step 2 of the algorithm we adopt the conventional rule that $LB^P = \infty$ if $(\Omega_1 \cap \Omega_2^P \cap \Omega_3^P) = \emptyset$.

3.5 Main Properties of the Algorithm

In this section the main properties of the branch and bound algorithm are briefly discussed.

Assume that $PR1^P$ is chosen as a subproblem. As it is clear from the discussion in the previous sections, these subproblems solved at nodes of the branch and bound tree are linear programs. Furthermore, a subproblem at a given node differs from the subproblem at its parent node only in certain coefficients of one (capacity) constraint. As a result of this, the solution to the linear program at the parent node may often be used as a starting solution for the linear subproblem at the successor node. Since this solution is not usually primal and/or dual feasible at the successor node, a primal-dual simplex procedure may be used for reoptimization. We will comment more on this in chapter 5.

Now assume that $(PRC1^P)$ is chosen as a subproblem, which is also a linear program. Let x_{sq} such that $s \in I_k$ be selected as a separation variable at node N^P at stage v . If $P_{kq}^P \neq \emptyset$ prior to the selection of x_{sq} then a subproblem at node N^P

differs from the subproblem of its parent node only in certain coefficients of one (capacity) constraint which corresponds to q^{th} period. So, the basis at the parent node may be utilized for the reoptimization of the subproblem at node N^p . In the case where $P_{kq}^p = \emptyset$ prior to the selection of a separation variable the basis of the subproblem at the parent node of N^p may still be used for reoptimization. If (s,q) is labeled "positive" at node N^{2v} then all of the constraints but two of them which are used to linearize the convex underestimating functions Δ_{sq}^p and v_{sq}^p are dropped from the constraint set of all of the subsequent subproblems generated at the successor nodes of node N^{2v} . Since some constraints are dropped from the linear program the basis at node N^p may not be used at node N_{2v} . If (s,q) is labeled "zero" at node N^{2v-1} then the coefficients of the variable x_{sq} are adjusted accordingly such that x_{sq} will never enter the basis. Since only the coefficients of one variable in some constraints are adjusted, the basis at node N^p may be used for solving the subproblem at node N^{2v-1} . We will comment more on the structure of the subproblem at node N^p and the reoptimization possibilities for (PRC1^p) in chapter 5.

As indicated in the previous sections, a feasible solution for (P_1) may be (usually will be) produced earlier in the algorithm. As a consequence of this, one may terminate the algorithm with an ϵ -optimum solution where ϵ is a tolerance level determined by the decision maker.

We are now in a position to show that the branch and bound algorithm we developed finds an optimal solution in a finite number of stages. To prove this, we need the following lemma and its corollary.

Lemma 2. Let N^p , $p=1,2,\dots,2^{T^*}$ be a terminal node. For any $(x,y) \in (\Omega_1 \cap \Omega^{p'}_2 \cap \Omega^p_3)$ the following statements are true

- a. $\Gamma^p_t(x) \geq \tau_t(x)$ for all $t \in \{1,2,\dots,T\}$
- b. $\sigma^p(x,y) \geq f(x,y)$

Proof. Since Γ and σ are similar in nature we will only prove part a of the lemma. Let N^p be any terminal node, t be any period, i be any family and $x^i_t \in (\Omega_1 \cap \Omega^{p'}_2 \cap \Omega^p_3)$. We will prove the theorem by showing that $\Gamma^{p_{it}}_t(x^i_t) \geq \tau_{it}(x^i_t)$. This completes the proof since N^p , t and i are arbitrarily chosen elements of $\{1,2,\dots,2^{T^*}\}$, $\{1,2,\dots,T\}$ and $i \in \{1,2,\dots,m\}$, respectively. Since N^p is the terminal node, $P^p_{it} = \emptyset$. If $P^p_{it} = \emptyset$ then $\Gamma_t(x) = \tau_t(x) = 0$. Therefore we will assume that $P^p_{it} \neq \emptyset$. There are two possible cases concerning P^p_{it}

Case 1. For all $j \in P^p_{it}$, $0 < x_{jt} \leq U_{jt}$

Case 2. There exist $j \in P^p_{it}$ such that $x_{jt} = 0$.

Case 1. Since for all $j \in P^p_{it}$, $0 < x_{jt} \leq U_{jt}$,

$$\Gamma_{it}(x^i_t) = S_i + \sum_{i=1}^m \sum_{j \in P^p_{it}} V_{jt}(x_{jt})$$

τ_{it} is given by,

$$\tau_{it}(x^i_t) = S_i + \sum_{i=1}^m \sum_{j \in P^p_{it}} V_{jt}(x_{jt})$$

Therefore $\Gamma_{it}(x^i_t) = \tau_{it}(x^i_t)$.

Case 2. Let $PP_{it}^p = \{j \in P_1: x_{jt} = 0\}$ and $PP_{it}^p = \{(j, t) \in P_{it}^p: x_{jt} > 0\}$. Then

$$\Gamma_{it}^p(x^i_t) = S_i + \sum_{j \in PP_{it}^p} (s_j + a_j x_{jt}) + \sum_{j \in PZ_{it}^p} (s_j + a_j x_{jt})$$

However $\tau_{it}(x^i_t) = S_i + \sum_{j \in (PP_{it}^p)} V_{jt}(x_{jt})$. Therefore, $\Gamma_{it}^p(x^i_t) \geq \tau_{it}(x^i_t)$.

The following corollary is an immediate consequence of lemma 2 and the fact that $(\Omega_1 \cap \Omega_2) \supset (\Omega_1 \cap \Omega_2')$.

Corollary 6. $(\Omega_1 \cap \Omega_2' \cap \Omega_3) \subset (\Omega_1 \cap \Omega_2 \cap \Omega_3)$ for all terminal nodes N^p , $p = 1, 2, \dots, 2^{rT}$.

Theorem 7. The branch and bound algorithm finds an exact optimum solution for (P_1) in a finite number of stages.

Proof. From the validity of the upper and lower bounds, and from the fact that no feasible solution $(x, y) \in (\Omega_1 \cap \Omega_2 \cap \Omega_3)$ is ever excluded from consideration, it follows that if the branch and the bound algorithm terminates at some stage v , then the incumbent solution $(x, y)_{UB}$ is an optimal solution for problem (P_1) .

The algorithm terminates when the candidate list becomes empty. Let C^v be the set of all nodes in the candidate list at stage v of the algorithm. Then if $LB^p \geq UB$ for all $N^p \in C^v$, the candidate list becomes empty and the algorithm terminates. At each stage of the algorithm a free variable x_{jt} is chosen and two new nodes are generated. Therefore, at most 2^{rT} terminal nodes can be created. If the candidate list becomes empty before all the terminal nodes are created, then the

algorithm terminates and is thus clearly finite. Therefore, to prove the finiteness part of the theorem we now show that if all 2^{rT} terminal nodes are created, then after the last terminal node is created the candidate list will become empty.

Assume that all 2^{rT} terminal nodes have been generated. Also without loss of generality assume that the candidate list contains these 2^{rT} nodes (some of these nodes may have been fathomed). Corollary 6 implies that $(\Omega_1 \cap \Omega^{p'}_2 \cap \Omega^p_3) \subset (\Omega_1 \cap \Omega_2 \cap \Omega_3)$ for all terminal nodes N^p , $p = 1, 2, \dots, 2^{rT}$. Let $(x^k, y^k) \in (\Omega_1 \cap \Omega^{p'}_2 \cap \Omega^p_3)$ be such that $\min_{p=1, 2, \dots, 2^{rT}} \sigma(x^p) = \sigma(x^k)$.

Furthermore, the branching rule guaranties that

$$\bigcup_{p=1}^{r_i} (\Omega_1 \cap \Omega^{p'}_2 \cap \Omega^p_3) = (\Omega_1 \cap \Omega_2 \cap \Omega_3). \quad \text{Let } (x^*, y^*) \text{ be an}$$

optimal solution for (P_1) . Then from the facts given above

$$\text{and from lemma 2b, } f(x^k, y^k) \leq \sigma(x^k) + \sum_{j=1}^r \sum_{t=1}^T h_j y^k_{jt} = \xi(x^k, y^k).$$

Since $f(x^k, y^k) \geq f(x^*, y^*)$, $\xi(x^k, y^k) \geq f(x^*, y^*) = UB$. Now let N^q be any terminal node. Clearly, $\xi(x^q, y^q) \geq f(x^q, y^q) \geq f(x^*, y^*) = f(x^p, y^p) = UB$, and node N^q will be fathomed. Since N^q is an arbitrary element of the set of terminal nodes, the proof is complete. ■

3.6. A Modified Branch and Bound Algorithm for MFDLZ

MFDLZ is a special class of MFDLC in which $K_i = 0$ and $k_j = 0$ for all $i \in \{1, 2, \dots, m\}$ and $j \in I_1$. Therefore the term $\psi(x)$ drops from the objective function of (P_1) . We now restate MFDLZ as

(PZ_1) minimize $g(x, y)$ subject to $(x, y) \in (\Omega_1 \cap \Omega_2 \cap \Omega_3)$.

Note that $g(x, y)$ is a linear function, hence we do not need to underestimate $g(x, y)$. The branch and bound algorithm for solving (PZ_1) is the same as the branch and bound algorithm described in Section 3.4. The following minor modifications in the branch and bound algorithm are due to the linearity of $g(x, y)$.

1. Let $(PRZ1^p)$ and $(PRCZ1^p)$ be the linear subproblems at node N^p which are obtained by relaxing the constraints (6) using the linear underestimator and the convex underestimator, respectively. Note that $(PRZ1^p)$ is similar to $(PR1^p)$, and $(PRCZ1^p)$ is similar to $(PRC1^p)$ except the latter problems have underestimating objective functions. Therefore, we can use the related underestimating functions developed in Chapter 3.

2. If at any node N^p $(x^p, y^p) \in (\Omega_1 \cap \Omega_2 \cap \Omega_3)$ then that node is fathomed because (x^p, y^p) is the best possible objective function value that can be attained at that node.

3. At node N^0 , $(PRCZ1^0)$ has mT more variables and rT more constraints than $PRZ1^0$ as opposed to $2mT$ more variables and $T(2r-1)$ more constraints in $(PRC1^0)$. When the first item in

family i in period t is labeled positive one of these additional variables and r_i-1 of these additional constraints are removed from (PRCZ1^P). In other words (PRC1^P) has twice as many additional variables as (PRCZ1^P) to obtain a linear subproblem. Therefore, the application of the convex envelope as an underestimator becomes more attractive since the problem size is proportionally reduced.

3.7. Preprocessing

Preprocessing the problem data before an algorithm is implemented on nonconvex programming problems has been suggested as a way to help reduce the computational effort (see for example Von Roy and Wolsey, 1987). The objective of preprocessing is to eliminate as much of the nonlinearity (and/or nonconvexity) as possible and is to obtain a "tighter" formulation of the problem.

In problem (P_1) a major source of nonlinearity and (nonconvexity) is the family setup time function and family setup cost function. For example if we knew that in any optimal solution for (P_1) $\sum_{j \in I_v} x_{jp} > 0$ for some family $v \in \{1, 2, \dots, m\}$ and some period $p \in \{1, 2, \dots, T\}$, then $G_{vp}(x_p^v)$ equals S_v . In this case constraint (6) for period p in (P_1) is

$$\sum_{\substack{i=1 \\ i \neq v}}^m [G_{ip}(x_p^i) + \sum_{j \in I_i} V_j(x_{jp})] + \sum_{j \in I_v} V_{jp}(x_{jp}) \leq C_p - S_v \quad (47)$$

and ψ_p becomes

$$\sum_{\substack{i=1 \\ i \neq v}}^m [F_{ip}(x_p^i) + \sum_{j \in I_i} M_j(x_{jp})] + \sum_{j \in I_v} M_{jp}(x_{jp}) + S_v \quad (48)$$

and the underestimating functions μ_{vp}^k and f_{vp}^k for $k = 1, 2, \dots$ will be accordingly modified if $(PR1^p)$ is used as a subproblem. If $(PRC1^p)$ is used as a subproblem then certain constraints are removed from the constraint set. In a situation where $s_j = 0$ for all $j \in I$ and $\sum_{j \in I_i} x_{jp} > 0$ for all

$i \in \{1, 2, \dots, m\}$, constraint (6) for period p and ψ_p becomes linear. Consequently, one does not need to branch on any variable x_{jp} , $j \in I$. In summary, the objective of the preprocessing step is to start the branch and bound algorithm with as "tight" a relaxation of problem (P_1) as possible.

Consider the following linear program for each $i \in \{1, 2, \dots, m\}$ and $t \in \{1, 2, \dots, T\}$:

$$[Pl(i, t)] \text{ minimize } z_{it} = \sum_{j \in I_i} x_{jt}$$

$$\text{s. t. } (x, y) \in (\Omega_1 \cap \Omega_2^0 \cap \Omega_3^0)$$

$$v^0(x, y) \leq UB$$

where UB is the upper bound which may be obtained from a heuristic procedure. In this chapter we assume upper bound is available. In the next chapter we develop a heuristic procedure from which UB can be obtained. In the preprocessing

step we first solve $P1(1,T)$. If $z_{1T} > 0$, then in any optimal solution for $(P_1) \sum_{j \in I_1} x_{jt} > 0$. Therefore, we first modify

constraint (6) for period T and ψ_T as we discussed above and then modify μ_{1T}^0 or ρ_{1T}^{020} accordingly. If $z = 0$, no adjustment can be done. Subsequently we move to problem $P1(2,T)$ and repeat the above procedure. We work our way back to problem $P1(m,1)$ in the manner explained above. In summary it takes mT linear programs to complete the preprocessing step. As will be seen in the next section solving mT linear programs is sometimes a relatively minor fraction of the total computational effort required by the branch and bound algorithm. This is especially true since any two linear programs that are solved successively are very similar to each other and the optimal solution for the first one can be used as a starting solution for the second.

Since the constraint sets of (P_1) and (PZ_1) are identical the only difference in preprocessing the data of (PZ_1) is in the additional constraint used in $P1(i,t)$. We now give the linear program that is used for preprocessing the data for MFDLZ. For each $i \in \{1, 2, \dots, m\}$ and $t \in \{1, 2, \dots, T\}$:

$$[PZ1(i,t)] \text{ minimize } z_{it} = \sum_{j \in I_1} x_{jt}$$

$$\text{s. t. } (x,y) \in (\Omega_1 \cap \Omega_2^{0'} \cap \Omega_3^0)$$

$$g^0(x,y) \leq UB$$

CHAPTER 4

HEURISTIC PROCEDURES FOR MFDLC AND MFDLZ

Florian et al., 1980 proved that a special case of MFDLC and MFDLZ is NP-hard problem. Therefore, it may be very difficult to solve realistic size MFDLC or MFDLZ problems optimally. A practical objective would be to develop heuristic procedures which provide "good" solutions. As remarked earlier, if a feasible solution for (P_1) or (PZ_1) is available then UB is set equal to the value of this solution, and the solution is stored in $(x,y)_{UB}$ as the incumbent in the initialization step of the branch and bound algorithm. Hence a heuristic procedure may help cut down the computational effort required by the branch and bound algorithm. We now give two additional properties for MFDLC and MFDLZ which will help develop the heuristic procedure. These properties are obvious and therefore, do not require proofs.

Property 5. Consider the following relaxation of problem (P_1) .

(PR_1) minimize $f(x,y)$, subject to $(x,y) \in (\Omega_1 \cap \Omega_2)$.

Let (x',y') be an optimal solution to (PR_1) . If (x',y') is feasible in (P_1) then (x',y') is an optimal solution for (P_1) .

Property 6. Let (x'', y'') be such that $x''_{jt} = d_{jt}$ and $y''_{jt} = 0$ for all $j \in I$ and $t \in \{1, 2, \dots, T\}$. If (x'', y'') is feasible in (PZ_1) , then (x'', y'') is an optimal solution for (PZ_1) .

We use properties 5 and 6, in finding starting solutions for the heuristic procedures for MFDLC and MFDLZ, respectively. We will however, defer discussion of finding starting solutions to section 4.2. In the following subsections we give the description of the heuristic procedures for MFDLC and MFDLZ. These procedures are similar except for their starting solutions and the manner the costs are evaluated. For this reason, we will only develop the heuristic procedure for MFDLC and state the modifications needed in this procedure to obtain the heuristic procedure for MFDLZ.

4.1. Description of the Heuristic Procedures

As stated in property 5, if for all $t \in \{1, 2, \dots, T\}$ the capacity in period t is sufficient to produce x'_{jt} for all $j \in I$, then the optimal production schedule is obtained. In most cases however, such a production schedule will violate the capacity constraints in certain periods and therefore, will not be feasible. The idea behind the heuristic procedure is to eliminate these capacity violations by shifting the production of certain items to earlier periods where there is

excess capacity. For any production schedule $x = \{x_{jt} : j \in I, t \in \{1, 2, \dots, T\}\}$ deficiency in a period is defined as the amount of capacity units by which the capacity constraint (6) for that period is violated.

Let x'_{jt} , $j \in \{1, 2, \dots, r\}$ and $t \in \{1, 2, \dots, T\}$, be defined as in property 5. To start the heuristic procedure let $x_{jt} = x'_{jt}$ for all $j \in I$ and $t \in \{1, 2, \dots, T\}$. If this schedule does not violate any of the constraints (6), then it is an optimal schedule. Therefore, the procedure terminates. However, usually there is at least one period t , $t > 1$ in which the capacity constraint is violated. Let $t' \in \{2, \dots, T\}$ be the largest period index where there is deficiency, that is, for all $t > t'$ constraint (6) is satisfied for the above schedule. Let x^* denote the lowest cost schedule obtained by the heuristic. The heuristic procedure sets $x^*_{jt} = x'_{jt}$, $j \in I$, $t \in \{t'+1, t'+2, \dots, T\}$. The deficiency in period t' can be eliminated by shifting a part or all of the lot sizes of some products to period $t'-1$. We call "a shift alternative", the choice of specific items along with their quantities to be shifted from period t to period $t-1$. Since the quantity of items transferred from period t to period $t-1$ represents the inventory carried from period $t-1$ to period t , the cost of a shift alternative can be computed easily. In period t' , first a set of shift alternatives are generated. Let $K(t')$ be the number of shift alternatives generated in period t' . The cost of the $K(t')$ alternatives are computed and ranked in ascending

order. Then from these $K(t')$ alternatives the first $k(t') \leq K(t')$ alternatives are chosen. Associated with each of the $k(t')$ alternatives is a partial (feasible) production schedule. Let $x(v, t')$ be the v^{th} partial schedule in period t' . Then for each $v \in \{1, 2, \dots, k(t')\}$, $x(v, t') = \{x_{jt}(v) : j \in I, t \in \{t', t'+1, \dots, T\}\}$ where x_{jt} , for all $j \in I$ is simply x'_{jt} , minus the number of units of item j shifted to period $t'-1$. The cost of the partial schedule $x(v, t')$ is simply the sum of the cost of the inventory carried from period $t'-1$ to t' and the associated joint and individual setup costs in period $t'-1$. Let $C[(v, t'), (p, t'+1)]$ denote the cost of partial schedule $x(v, t')$. Here p is the index of the shift alternative in period $t'+1$ from which shift alternative v in period t' is generated. Since there is no deficiency in period $t'+1$, the only shift alternative generated in period $t'+1$ would be not to shift any product from period $t'+1$ and therefore $C[(1, t'+1), (1, t'+2)] = 0$. Associated with each of the $k(t')$ alternatives are the trial production quantities for period $t'-1$. Let $q_{j, t'-1}(v)$ be the trial production quantity in period $t'-1$ for each item $j \in I$ under the shift alternative $v \in \{1, 2, \dots, k(t')\}$ from period t' . Then for each $j \in I$, $q_{j, t'-1}(v)$ is simply $x'_{j, t'-1}$ plus the quantity of item j that was shifted to period $t'-1$ under the v^{th} shift alternative in period t' .

For any $v \in \{1, 2, \dots, k(t')\}$ for which the trial quantities $q_{j, t'-1}(v)$, $j \in I$ violate the capacity constraint a number of shift alternatives which eliminate the infeasibility in period

$t'-1$ are generated. These shift alternatives will be explained in the subsequent sections. Any $v \in \{1, 2, \dots, k(t')\}$ for which the trial quantities do not violate the capacity constraint in period $t'-1$, results in only one shift alternative in period $t'-1$. This alternative is not to shift any production to period $t'-2$. Let $K(t'-1)$ be the total number of shift alternatives generated in period $t'-1$. These shift alternatives are ranked in ascending order of their costs, $C[(v, t'-1), (p, t')]$ and the first $k(t'-1) \leq K(t'-1)$ of the alternatives are chosen. Again associated with the v^{th} shift alternative in period $t'-1$, where $v \in \{1, 2, \dots, k(t'-1)\}$ there is a partial feasible schedule $x(v, t'-1) = \{x_{jt}(v) : j \in I, t \in \{t'-1, t', \dots, T\}\}$. This process is repeated until $t = 2$ and the complete feasible schedule $x(v, 1) = \{x_{jt}(v), q_{j,1}(v) : j \in I, t \in \{1, 2, \dots, T\}\}$ with the minimum cost $C[(v, 2), (p, 3)]$ among the $K(2)$ schedules is chosen as the "best solution" generated by the heuristic procedure. The development of the heuristic procedure for MFDLZ is similar to that for MFDLC with the exception that x'_{jt} is replaced with d_{jt} for all $t \in \{1, 2, \dots, T\}$ and $j \in I$, and joint and individual setup costs are not included in computing the cost of a partial schedule since these costs are zero.

We now give the following additional notation. For any alternative $v \in \{1, 2, \dots, K(t)\}$ and $t \in \{1, 2, \dots, T\}$:

$$L[(v,t),(p,t+1)] = \begin{cases} 1 & \text{if the } v^{\text{th}} \text{ shift alternative} \\ & \text{in period } t \text{ follows from the} \\ & p^{\text{th}} \text{ shift alternative in period} \\ & t+1 \text{ where } v \in \{1,2,\dots,K(t)\}, \\ & p \in \{1,2,\dots,K(t+1)\}, \\ & t \in \{2,3,\dots,t'\}. \\ 0 & \text{otherwise} \end{cases}$$

$C[(v,t),(p,t+1)]$: Cost of the partial production schedule $x(v,t)$ which followed from the p^{th} shift alternative in period $t+1$, $v \in \{1,2,\dots,k(t)\}$, $p \in \{1,2,\dots,k(t+1)\}$, $t \in \{2,3,\dots,t'\}$.

$z_j(v,t)$: Number of units of item j ; $j \in I$ whose production was shifted from period t to period $t-1$ under the v^{th} shift alternative in period t .

$Q_t(v)$: The amount of deficiency (in units of capacity) in period t under the shift alternative v .

$$I_{it}^+(v) = \{j \in I_i : q_{jt}(v) > 0\}.$$

$$F_t(v) = \{i \in \{1,2,\dots,m\} : I_{it}^+(v) \neq \emptyset\}.$$

Then in period t we have the following relation between $Q_t(v)$ and $q_{jt}(v)$ for all $j \in I$.

$$Q_t(v) = \max \{0, ([\sum_{i \in F_t(v)} S_i + (\sum_{j \in I_{it}^+(v)} [s_j + a_j q_{jt}(v)])] - C_t)\} \quad (49)$$

For each $i \in \{1,2,\dots,m\}$ and each $t \in \{1,2,\dots,T\}$ let $q_{jt}^i(v) = \{q_{jt}(v) : j \in I_i\}$. Then for any shift alternative $v \in \{1,2,\dots,K(t)\}$ and for any shift alternative $p \in \{1,2,\dots,k(t+1)\}$, $C[(v,t),(p,t+1)]$ is given by

$$C[(v,t), (p,t+1)] = \begin{cases} \sum_{j \in I} h_j z_j(v,t) & \text{if } L[(v,t), (p,t+1)] = 1 \\ + \sum_{i=1}^m \psi_{it}(q_t^i(v)) & \\ + C[(p,t+1), (w,t+2)] & \\ \infty & \text{otherwise} \end{cases}$$

To initiate the heuristic procedure we set

for all $t > t'$

$$K(t) = k(t) = 1$$

$$L[(1,t), (1,t+1)] = 1$$

$$C[(1,t), (1,t+1)] = 0 \text{ for all } t > t' + 2 \text{ and}$$

$$C[(1,t'+1), (1,t'+2)] = \sum_{j=1}^r \sum_{t=1}^T h_j y'_{jt}$$

and for all $t \in \{1, 2, \dots, t'\}$, for all $j \in I$, and for all $v \in \{1, 2, \dots, K(t)\}$ $x_{jt}(v) = x'_{jt}$.

The following modifications are made in the heuristic procedure for MFDLZ.

1. For all $t \in \{1, 2, \dots, T\}$ and all $j \in I$ x'_{jt} is replaced with d_{jt} .

2. The following term is set to zero

$$\sum_{i=1}^m \psi_{it}(q_t^i(v))$$

3. $C[(v,t'), (1,t'+1)] = 0$.

We should note that in each period where there is deficiency, the number of shift alternatives one can generate to eliminate the infeasibility is in general infinite. We

consider two types of alternative generation schemes: individual item release scheme and family release scheme which are discussed in the following sections.

We will use the following example to illustrate the heuristic procedure.

Example 1. Consider four period, six item and two family production planning problem. The demand for each item $j \in \{1, 2, \dots, 6\}$ in each period $t \in \{1, 2, \dots, 4\}$ is given in table 1. Individual setup cost, individual setup time, capacity absorption rate and cost of holding inventory per unit per period are given in table 2. For each family $i \in \{1, 2\}$, setup cost and setup time are given in table 3. Let items one, two and three be in family one and items four through six be in family two.

Table 1. Demand data for a six-item four-period two-family problem

Item	Demand (d_{jt})			
	Period 1	Period 2	Period 3	Period 4
1	53	8	72	68
2	25	88	35	85
3	0	198	34	0
4	12	138	108	101
5	4	88	39	42
6	22	46	83	10
Capacity	1196	1875	1090	1100

Table 2. Individual setup time, individual setup cost, capacity absorption rate and individual holding cost data

Item	Ind. setup time	Ind. setup cost(\$)	cap. abs. rate	inv. hold. cost
1	10	0.0	3	5.19
2	14	0.0	2	4.14
3	6	0.0	1	3.28
4	12	0.0	2	3.76
5	18	0.0	4	3.14
6	25	0.0	4	3.41

Table 3. Setup cost and time data

Family	Setup cost (\$) K_i	Setup time S_i
1	0.0	168
2	0.0	249

4.1.1. Individual Item Release Scheme

An individual item release scheme considers the products independent of their families. In this approach, items are ranked in ascending order of their h_j/a_j value. Let $\rho(w)$, $w \in \{1, 2, \dots, r\}$ be the product index with the w^{th} smallest h_j/a_j ratio. We can generate r different alternatives by using this scheme in the following manner.

Let $w \in \{1, 2, \dots, r\}$ be an alternative that will be generated in period t by using individual item release scheme. We will refer to this alternative as w^{th} minimum alternative. Let v' be an alternative generated in period $t+1$ from which alternative w is generated. Let $k=1$ be the first attempt to eliminate a deficiency in period t for alternative w . In the individual item release scheme, the first item whose production is shifted from period t to period $t-1$ is $\rho(w)$. The values of $z_{\rho(w)}(w, t)$, $q_{\rho(w)t-1}(w)$ and $x(\rho(w), t)$ are given by,

$$z_{\rho(w)}(w, t) = \begin{cases} q_{\rho(w)t}(v') & \text{if } Q_t(v') \geq A_1 \text{ and for some } j \in \{I_{\rho(w)t} \setminus \rho(w)\} \text{ } q_{jt}(v') > 0 \\ & \text{or } A_1 \geq Q_t(v') \end{cases} \quad (50a)$$

$$q_{\rho(w)t}(v') & \text{if } Q_t(v') \geq A_2 \text{ and } q_{jt}(v') = 0 \\ & \text{for all } j \in \{I_w \setminus \rho(w)\} \\ & \text{or } A_2 \geq Q_t(v') \end{cases} \quad (50b)$$

$$Y \quad \text{otherwise} \quad (50c)$$

where

$$A_1 = q_{\rho(w)t}(v') a_{\rho(w)} + s_{\rho(w)}$$

$$A_2 = q_{\rho(w)t}(v') a_{\rho(w)} + s_{\rho(w)} + S_{\rho(w)}$$

$$Y = \max\{0, Q_t(v')/a_{\rho(w)}\}$$

$$q_{\rho(w)t-1}(w) = x_{\rho(w)t-1} + z_{\rho(w)}(w, t) \quad (51)$$

$$x_{\rho(w)t}(w) = \max\{0, (q_{\rho(w)t}(w) - z_{\rho(w)}(w, t))\} \quad (52)$$

After the production of $\rho(w)$ is shifted k is set to $k+1$.

The remaining deficiency in period t is given by $Q_t^k(v')$,

$$Q_t^k(v') = \begin{cases} \max\{0, [Q_t^{k-1}(v') - (x_{\rho(w)t}(w) a_{\rho(w)} + s_{\rho(w)})]\} \\ \text{if condition in 50a holds} \end{cases} \quad (53a)$$

$$\begin{cases} \max\{0, [Q_t^{k-1}(v') - (x_{\rho(w)t}(w) a_{\rho(w)} + s_{\rho(w)} + S_{\rho(w)})]\} \\ \text{if condition in 50b holds} \end{cases} \quad (53b)$$

$$0 \quad \text{otherwise} \quad (53c)$$

where $Q_t^1(v') = Q_t(v')$ which is computed from (49).

If $Q_t^k(v') = 0$ then the alternative generation scheme is completed. On the other hand if $Q_t^k(v') > 0$ then additional items need to be shifted from period t . Suppose that $Q_t^k(v') > 0$. Let the item to be shifted in the $(k+1)^{st}$ attempt be $\rho(b)$. Let $FL_k(w)$ be the index set of products whose production were shifted prior to the $(k+1)^{st}$ attempt under the w^{th} shift alternative. Then $\rho(b)$ is given by

$$\rho(b) = \begin{cases} \rho(k) & \text{if } \rho(k) \notin FL_k(w) \\ \rho(k+1) & \text{otherwise} \end{cases}$$

The values of $z_{\rho(w)}(w, t)$, $q_{\rho(w)t-1}(w)$ and $x_{\rho(w)t}(w)$ are computed from equations (50), (51) and (52), respectively. Here, $\rho(w)$ in (50), (51) and (52) is replaced with $\rho(b)$. $Q_t^k(v')$ is computed from (53). This procedure is repeated until $Q_t^1(v') = 0$, where $1 \leq l \leq r$, that is until the deficiency is completely eliminated. Let the deficiency be completely eliminated in the l^{th} attempt. Then we set

$$x_{\rho(w)t}(w) = q_{\rho(v'), t}(v') \quad \text{for all } \rho(w) \geq l+1$$

$$z_{\rho(v')}(w, t) = 0 \quad \text{for all } \rho(w) \geq l+1$$

We now give the following obvious result. Hence the proof is omitted.

Example 2. We will use the data of example 1 and generate the first two alternatives in period 4 by using individual item release scheme. First we give the initial

conditions

$$K(5)=k(5)=1$$

$$L[(w,5), (1,6)]=1 \quad \text{for all } w=1,2$$

$$C[(w,5), (1,6)]=0.0 \quad \text{for all } w=1,2$$

$$q_{j4}(w)=d_{j4} \quad \text{for all } j \in I \text{ and } w=1,2$$

We also compute h_j/a_j for all $j \in I$. They are given as;

$$h_5/a_5=0.79, \quad h_6/a_6=0.85, \quad h_1/a_1=1.73, \quad h_4/a_4=1.88, \quad h_2/a_2=2.07, \\ h_3/a_3=3.28.$$

The deficiency in period 4, $Q_4(1)=186$, is computed from (49). Therefore, any shift alternative should save at least 186 units of capacity. Let's generate the first two alternatives by using alternative generating scheme. Let $w=1$. Since $w=1$, $\rho(1)=5$. Therefore, the first item to be released is 5. $q_{54}(1) \cdot (a_5)=168$. Item 5 will be completely shifted. Therefore, $z_5(1,4)=42$ and $Q_4^1(1)=186-(168+18)=0$. Since $Q_4^1(1)=0$, this shift alternative is generated. We then compute the following quantities,

$$q_{53}(1)=39+42$$

$$q_{j3}(1)=d_{j3} \quad \text{for all } j \in I, j \neq 5.$$

$$x_{54}(1)=0$$

$$x_{j4}(1)=d_{j4} \quad \text{for all } j \in I, j \neq 5.$$

$$L[(1,4), (1,5)]=1$$

$$C[(1,4), (1,5)]=0.0+(42 \cdot 3.14)=131.88.$$

Let $w=2$. $\rho(2)=6$. $q_{64}(2)(a_6)=40$. Item 6 will completely be shifted. Therefore, $z_6(2,4)=10$ and $Q_4^1(1)=186-(40+25)=121$. $\rho(1)=5$. So item 5 will next be shifted. $z_5(2,4)=121/4=30.25$

and $Q_4^2(1)=121-(30.25*4)=0$. Since $Q_4^2(1)=0$, this shift alternative is also generated. We then compute the following quantities,

$$q_{63}(2)=83+10=93.$$

$$q_{53}(2)=39+30.25=69.25.$$

$$q_{j3}(2)=d_{j3} \quad \text{for all } j \in I, j \neq 5 \text{ and } 6.$$

$$x_{64}(2)=0.0$$

$$x_{54}(2)=42-30.25=11.75$$

$$x_{j4}(2)=d_{j4} \quad \text{for all } j \in I, j \neq 5 \text{ and } 6.$$

$$L[(2,4), (1,5)]=1$$

$$C[(2,4), (1,5)]=0.0+(10*3.41)+(30.25*3.14)=129.93.$$

4.1.2. Family Release Scheme

The next method for generating shift alternative is the family release scheme. In this scheme, a shift alternative is generated by shifting the total production of all the items in a set of families from period t to period $t-1$. Let B be a set of families whose items will be shifted from period t to $t-1$. Note that, B may contain one or more families and the production quantities of all items whose families are in B are shifted from period t to period $t-1$. If there still remains some deficiency in period t after the production quantities of all items whose families are in B is shifted, the remaining deficiency is iteratively eliminated by shifting the production of the item which has $\rho(k)$ value. Here k iterates

from 1 through r provided that $I_{\rho(k)} \notin B$. If $I_{\rho(k)} \in B$ then k is incremented by one without any shift.

Note that for each set only one alternative is generated. The number of alternatives that can be generated with the family release scheme depends on the number of different sets taken into consideration. We will comment on the possible choices of B in discussing the computational results. We will now discuss certain properties of a shift alternative under the family release scheme and then we will give the necessary formulas and procedures for generating this alternative. For any $t \in \{1, 2, \dots, T\}$ and any shift alternative w , we need to give the following additional notation.

$$I_{Bt}^+(w) = \{j : j \in I_1, i \in B, \text{ and } q_{jt} > 0\}$$

$$F_{Bt}(w) = \{i : \text{for some } j \in I_1, j \in I_{Bt}^+(w)\}$$

A: any subset of B such that $B \setminus A$ contains only one family.

We are now in a position to generate the shift alternative in period t for set B using the family release scheme. Let w be this alternative. Also let v be the a shift alternative in period $t+1$ from which w is generated. We will give the formulas for $z_j(w, t)$, $q_{jt-1}(w)$ and $x_{jt}(w)$. For all $j \in I_{Bt}$ these formulas are given as follows,

$$z_j(w, t) = q_{jt}(v)$$

$$q_{jt-1}(w) = z_j(w, t) + x_{jt-1}.$$

$$x_{jt}(w) = 0.0$$

Let $k = 0$, and $Q_t^k(v)$ be the remaining deficiency after all the production of items whose families are in B is shifted from period t to period $t-1$. Then

$$Q_t^{k+1}(v) = \max \{0, Q_t^k(v) - (\sum_{i \in F_{Bt}(v)} S_i + \sum_{j \in I_{Bt}^+} S_j + q_{jt}(v)a_j)\}$$

where $Q_t^0(v)$ is computed from (49).

If $Q_t^{k+1}(v) = 0$, w is generated. The procedure moves on to another set to generate another shift alternative. If $Q_t^{k+1}(v) \neq 0$ there still remains some deficiency. To eliminate the remaining deficiency individual item release scheme is applied in the following manner. Let b be the smallest integer such that $\rho(b) \in I_{Bt}^+$. First k is set to $k+1$. The first item to be shifted among the individual items is $\rho(b)$. For this purpose (50), (51) and (52) are used to compute $z_{\rho(b)}(w, t)$, $q_{\rho(b)t-1}(w)$ and $x_{\rho(b)t}(w)$, respectively. Then $Q_t^k(v)$ is updated by using (52). If $Q_t^k(v) \leq 0$, procedure stops. Otherwise b is set to $b+1$ and procedure is repeated until all deficiency is eliminated in this manner.

We now use the data given in example 1 and one of the shift alternatives generated in example 2 to illustrate the generation of shift alternatives by using family release scheme.

Example 3. For the purpose of illustration assume that we will generate an alternative in period 3 by using the first alternative of period 4. We first calculate the total deficiency in period 3. We can calculate $Q_3(1)$ using the data

of example 2 and equation (49). $Q_3(1)=1682-1090=592$. Assume that we will release family 1 in period 3 (i.e., we will shift all the production of items 1, 2 and 3). The total savings in capacity units in this case is 518 units. The remaining deficiency is $Q_3^1(1)=592-518=74$. To eliminate this deficiency we will use the individual item release scheme. Since $\rho(1)=5$. So item 5 will next be shifted. $z_3(1,3)=74/4=18.5$ and $Q_3^1(1)=0$. Since $Q_3^1(1)=0$, this shift alternative has been generated. We then compute the following quantities,

$$q_{12}(1)=8+72=80$$

$$q_{22}(1)=88+35=128$$

$$q_{32}(1)=138+108=246$$

$$q_{52}(1)=88+18.5=106.5$$

$$q_{j2}(1)=d_{j2} \quad \text{for all } j=4,6$$

$$x_{13}(1)=0.0$$

$$x_{23}(1)=0.0$$

$$x_{33}(1)=0.0$$

$$x_{53}(1)=81-18.5=62.5$$

$$x_{j3}(1)=d_{j3} \quad \text{for all } j=4,6$$

$$L[(1,3), (1,4)] = 1$$

$$C[(1,3), (1,4)] = 131.88 +$$

$$(72*5.19 + 35*4.14 + 34*3.28 + 18.75*3.14) \\ = 688.98$$

Lemma 3. Let $w \in \{1, 2, \dots, K(2)\}$. If $Q_1(w) > 0$ then the production schedule which is obtained by using any shift

alternative is infeasible.

The following example illustrates the generation of shift alternatives under individual item release scheme.

4.2. Finding A Starting Solution

As explained in the development of the heuristic procedure, the procedure starts with a starting solution and establishes initial conditions. Starting from the last period shift alternatives are generated in period t using the individual item release scheme and the family release scheme. For each shift alternative the algorithm computes the cost of the associated partial production schedule and ranks the alternatives in ascending order of their cost values. From these alternatives $k(t) \leq K(t)$ are selected. The algorithm then moves to period $t-1$. For each of these $k(t)$ alternatives, shift alternatives in period $t-1$ are generated in the same manner. The algorithm terminates when it reaches the first period.

A starting solution to the heuristic procedure for MFDLZ can directly be obtained from property 6. A starting solution to the heuristic procedure for MFDLC can be obtained by using property 5. In the literature the problem (PR_1) given in property 5 is known as multiproduct dynamic lot sizing problem with coordinated replenishments. Erenguc (1988), Kao (1979), Silver (1979), Veinot (1969) and Zangvill (1966) proposed

solution procedures for this problem. Among these algorithms, the algorithm developed by Erenguc (1988) appears to be the most efficient. When capacity constraints are not present to the problem, determination of lot sizes of the products in each family is independent of the lot sizes in the other families. Consequently, the algorithm given in Erenguc (1988) considers only one family. To obtain a starting solution for the heuristic procedure, we first run this algorithm for each family i , $i \in \{1, \dots, m\}$, establish initial conditions and then for each period we generate the alternatives using the individual item release scheme and the family release scheme starting from the last period.

4.3. Eliminating Inferior Alternatives in the Heuristic Procedure For MFDLZ

Since setup costs are not incurred in MFDLZ it is possible to eliminate from consideration certain alternatives generated by using the family release scheme. Suppose that there is a shift alternative v in period t which only shifts the production of some items whose families are in B . Furthermore, suppose that in this alternative there exists an item $j \in I_{Bt}^+$ such that its trial quantity is not completely shifted from period t to period $t-1$. This alternative is said to be a partial release alternative. Let v' be a shift alternative in period t generated by shifting the trial

quantities of all items whose families are in B. If there exists a shift alternative v as described above in any period then it is clear that partial release of the items whose families are in B can overcome the deficiency in period t . Furthermore, the cost of partial production schedule defined by shift alternative v is less than or equal to the cost of partial production schedule defined by shift alternative v' . Therefore v' is said to be an inferior alternative since this alternative will be dominated by at least one other shift alternative in period t , namely by v .

In generating an alternative v by using the family release scheme, we first check whether partial release of the items whose families are in B can overcome the deficiency. If the partial release of these items can overcome the deficiency we eliminate v from further consideration and we say v is an inferior alternative. Note that in the heuristic procedure we do not generate v' which dominates v . Neither we are assured that v' is ever generated. But, if the number of alternatives generated in any period is considered one may eliminate the inferior alternative for practical purposes. Also, the partial costs of the some of the alternatives generated are usually lower than that of the cost of an inferior alternative.

4.4. Feasibility in the Heuristic Procedure

The heuristic procedures given in the previous sections do not always guarantee a feasible schedule. Under certain conditions, however, it is possible to show that these heuristic procedures are guaranteed to generate a feasible schedule. Since the feasible region of MFDLC and MFDLZ are similar and the same alternative generating schemes are used feasibility conditions developed in this section apply to both algorithms.

As explained earlier, the procedure starts with setting x_{jt} to x'_{jt} (or d_{jt}) for all $j \in I$ and $t \in \{1, 2, \dots, T\}$. There may be some periods where the capacity is sufficient to produce x'_{jt} (or d_{jt}), $j \in I$ in that period. Should this case occur in period t , we say that there is excess capacity in period t and we denote the amount of excess capacity by E_t . Let $x_{jt} = x'_{jt}$ or d_{jt} , $j \in I$, $t \in \{1, 2, \dots, T\}$. To develop a formula for E_t the following notation and definitions are given. For each $i \in \{1, 2, \dots, m\}$ and for each $t \in \{1, 2, \dots, T\}$

$$I^+_{it} = \{j : j \in I_i \text{ and } x_{jt} > 0\}$$

$$F_t = \{i : \text{for some } j \in I_i \text{ } j \in I^+_{it}\}$$

Given the above notation we can compute E_t , $t \in \{1, 2, \dots, T\}$ as follows

$$E_t = \max \{0, [C_t - \sum_{i \in F_t} S_i + (\sum_{j \in I^+_{it}} S_j + a_j x_{jt})]\} \quad (54)$$

There may also some periods where there is deficiency. Otherwise optimal solution would have been obtained. Let Q_t denotes the deficiency in period t when $x_{jt} = x'_{jt}$ or d_{jt} for all $j \in I$ and $t \in \{1, 2, \dots, T\}$. Then Q_t , $t \in \{1, 2, \dots, T\}$ is given by

$$Q_t = \max \{0, [\sum_{i \in F_t} S_i + (\sum_{j \in I_{it}} s_j + a_j x_{jt}) - C_t]\} \quad (55)$$

Let $SE_t = \sum_{j=1}^t E_j$ and $SQ_t = \sum_{j=1}^t Q_j$, $t \in \{1, 2, \dots, T\}$. We now give

some sufficiency conditions for generating a feasible solution by using the heuristic procedure provided that for all $t \in \{1, 2, \dots, T\}$ $k(t) = K(t)$. The first condition considers a special case of MFDLZ where $d_{jt} > 0$ for all $j \in I$ and for all $t \in \{1, 2, \dots, T\}$, and it is given by the following lemma.

Lemma 4. Suppose that $d_{jt} > 0$ for all $j \in I$ and for all $t \in \{1, 2, \dots, T\}$. The heuristic procedure always generates a feasible schedule if the following condition holds.

$$SQ_t \leq SE_t \quad \text{for all } t \in \{1, 2, \dots, T\} \quad (56)$$

Proof. In computing Q_t and E_t all family setup times and individual setup times are included (equations (54) and (55)). Therefore, the maximum amount of deficiency transferred from period t to period $t-1$, in terms of capacity units is Q_t . Since family setup times and individual setup times are included in the computation of (55) for period $t-1$, maximum deficiency in period $t-1$ is $Q_{t-1} + Q_t$. The total maximum deficiency to be shifted in periods 1 through t is given by

$\sum_{j=1}^t Q_j$. Since it is assumed that $\sum_{j=1}^t E_j \geq \sum_{j=1}^t Q_j$, there exists a feasible solution to MFDLC. Furthermore, the heuristic procedure generates a feasible schedule because in any shift alternative maximum amount of production shift from period t to period $t-1$ in capacity units is Q_t . Also, for any shift alternative the maximum deficiency in period $t-1$ is $Q_{t-1} + Q_t$, since all products are scheduled to be produced in period $t-1$. Therefore, successive application of any shift alternative is capable of generating a feasible schedule as long as (56) holds. ■

Although Lemma 4 gives a sufficient condition for generating a feasible schedule, its underlying assumption is rather restrictive. For it is not always a common practice to observe demand for each product in each period. Lemma 4, however, gives an idea for generating a feasible schedule. We will use Lemma 4 in developing a sufficient condition for a more general case. This sufficient condition is given by Lemma 5.

Lemma 5. Suppose that $d_{jt} \geq 0$ for all $j \in I$, $t \in \{1, 2, \dots, T\}$ and for all $i \in \{1, 2, \dots, m\}$ $i \in F_1$. The heuristic procedure generates a feasible schedule if the following condition holds.

$$\sum_{k=1}^t Q_k \leq \sum_{k=1}^t RE_k = \sum_{k=1}^t [E_k - \sum_{i=1}^m \sum_{j \in I_{ik}^*} s_j] \text{ for all } t \in \{1, 2, \dots, T\} \quad (57)$$

Proof. Let t be any period. Maximum amount of production transferred from period t to period $t-1$ is given by

$$Q_t + \sum_{i \notin F_{t-1}} S_i + \sum_{i=1}^m \sum_{j \in I_{it-1}^+} S_j \quad (58)$$

Therefore, the total maximum deficiency in period $t-1$ is

$$Q_{t-1} + Q_t + \sum_{i \notin F_{t-1}} S_i + \sum_{i=1}^m \sum_{j \in I_{it-1}^+} S_j \quad (59)$$

It is, however, always possible to eliminate the effect of

$\sum_{i \notin F_{t-1}} S_i$ in (59) for period $t-1$. If $t = 2$ this effect is

eliminated because in period 1 at least one product in each family is assumed to have positive demand. If $t = 1$, (57) implies that the production schedule is feasible because $Q_1 = 0$. For $t \geq 3$, there is always an alternative generated by family release scheme that shifts the production of all items in family $i \notin F_{t-1}$. Therefore, production of these products can be shifted to period $t-2$. Hence, it eliminates this effect. If, in period $t-2$, for some $j \in I_i$ where $i \notin F_{t-1}$ $x_{jt} > 0$, then S_i term which must be included in period $t-1$ is eliminated. Otherwise, that is, if for some $i \notin F_{t-1}$, $i \notin F_{t-2}$, then production of the items in those families can again be shifted to period $t-3$. Therefore, in the worst case in period 2, total maximum deficiency will be

$$\sum_{j=k}^t Q_k + \sum_{k=2}^t \sum_{i=1}^m \sum_{j \in I_{ik}^+} S_j + \sum_{i \notin F_2} S_i$$

Since it is assumed that for all $i \in \{1, 2, \dots, m\}$, $i \in F_1$, total maximum deficiency in period 1 is given by

$$\sum_{k=1}^t Q_k + \sum_{k=1}^t \sum_{i=1}^m \sum_{j \in I_{ik}^+} S_j \quad (60)$$

Note that (60) is the total maximum deficiency and it is assumed that (57) holds. Therefore, total excess capacity is sufficient to eliminate the total deficiency. ■

As shown in the proof, there is always an alternative generated from family release scheme that will eliminate the maximum deficiency in any period. Therefore, the heuristic procedure generates a feasible schedule.

Note that RE_t , in (57) is the revised total excess capacity for any $t \in \{1, 2, \dots, T\}$ which is obtained by subtracting individual setup times for all products $j \in I_{it}^+$, for all $i \in \{1, 2, \dots, m\}$ and for all $t \in \{1, 2, \dots, T\}$. RE_t , $t \in \{1, 2, \dots, T\}$ can be used only for the transfer of production of some items in period t and for compensating family setup time, if required. In a family release scheme it is possible to generate alternatives for all possible combination of families. It is, therefore, always possible to shift the production of these families to an earlier period, which gives a flexibility.

We would like to note that it is possible to relax one of the assumptions of Lemma 5 which requires that in period 1 for all $i \in \{1, 2, \dots, m\}$ there exists at least one product j such that $d_{j1} > 0$. If this assumption is relaxed, (57) is

modified as follows: For all $t \in \{1, 2, \dots, T\}$

$$\sum_{k=1}^t Q_k \leq \sum_{k=1}^t RE_k = \sum_{k=1}^t [E_k - \sum_{i=1}^m \sum_{j \in I'_{ik}} S_j] - \sum_{i \in F_1} S_i \quad (61)$$

Lemma 4 and Lemma 5 state some sufficient conditions for generating a feasible solution in the heuristic procedure. We would like to note that these conditions are data independent conditions. Depending on the topology of the data, it is possible to develop problem dependent more restrictive conditions.

CHAPTER 5

COMPUTATIONAL STUDY

In this chapter we discuss the implementation of the preprocessing the data for (PZ_1) and (P_1) . Then we discuss other computational issues such as search strategy, selection of the separation variable and reoptimization. We also report computational results with the branch and bound algorithms for MFDLZ and MFDLC problems in this chapter. Two branch and bound algorithms were developed for MFDLC and MFDLZ. One of these algorithms uses linear underestimators for the subproblems at the nodes of branch and bound tree and the second algorithm uses convex envelopes for the subproblems. As will be seen, using convex envelopes in the branch and bound algorithm for (PZ_1) is computationally prohibitive. Since solving (P_1) is more difficult than solving (PZ_1) we choose not to implement the branch and bound algorithm for MFDLC that uses convex envelopes in the subproblems. However, we develop another underestimator which uses the convex envelope results. Computational results with the branch and bound algorithm whose subproblems are constructed with this underestimator are also discussed in this chapter. Finally, we discuss computational results for the heuristic procedures.

5.1. Preprocessing the Data

As explained in chapter 3, it is possible to preprocess the data by solving $P1(i,t)$ for each $i \in \{1,2,\dots,m\}$ and for each $t \in \{T,T-1,\dots,1\}$ successively. It takes mT linear programs to complete the preprocessing step. As will be seen in the next section, solving mT linear programs is sometimes a relatively minor fraction of the total computational effort required by the branch and bound algorithm. This is partly a consequence of the fact that any two linear programs that are solved successively are very similar to each other and the optimal solution for the first problem is used as a starting solution for the second. Naturally, once a family setup in a period is fixed then appropriate adjustments in the family and individual setup times and costs are made for the subsequent problems solved in the preprocessing stage. In implementing the preprocessing procedure for (P_1) we successively solve $P1(i,t)$ for each $t \in \{T,T-1,\dots,1\}$ and for each $i \in \{1,2,\dots,m\}$. We choose to solve $P1(i,t)$ starting with the highest t index because if a family is fixed at a later period the subsequent problems will have tighter constraints due to the dynamic nature of the problem.

In implementing the preprocessing procedure for (PZ_1) we solve $PZ1(i,t)$ for each $i \in \{1,2,\dots,m\}$ and $t \in \{T,T-1,\dots,1\}$. In order to make the relaxation tighter, in solving the linear program $PZ1(i,t)$, $i \in \{1,2,\dots,m\}$, $t \in \{T,T-1,\dots,1\}$ we use the

following property. Property 7 is obvious, therefore, its proof is omitted.

Property 7. Let t be any period. If $d_{jt} > 0$ for some product j , $j \in I_t$, for some $i \in \{1, 2, \dots, m\}$ such that $C_t > S_i + s_j$ then in any optimal solution for (PZ_1) $\sum_{j \in I} x_{jt} > 0$.

Property 7 states that if the capacity in period t exceeds the family setup time of some family $i \in \{1, 2, \dots, m\}$ plus the individual setup time of any product $j \in I_t$ which has a nonzero demand in period t , then there must be production in period t .

We implemented property 7 by setting the minor setup time $s_j \leftarrow s_j - \min_{j \in I} \{s_j\}$ for each $j \in I$, the joint setup time $S_i \leftarrow S_i - \min_{j \in I} \{S_i\}$, capacity $C_t \leftarrow C_t - \min_{j \in I} \{s_j\} - \min_{i \in \{1, 2, \dots, m\}} \{S_i\}$ for each $i \in \{1, 2, \dots, m\}$ where conditions of Property 7 were satisfied.

5.2 Other Computational Issues

Two branch and bound algorithms were implemented for solving (PZ_1) . In one of these algorithms we implemented the linear underestimators for the capacity constraints in the subproblem at each node N^p . In the second algorithm we implemented the convex envelopes for the capacity constraints in the subproblem at each node of the branch and bound tree. For solving (P_1) , we implemented only the branch and bound

algorithm whose subproblems were constructed by using linear underestimators. In all of these algorithms we used a depth-first (LIFO) search strategy. Since the depth of the branch and bound tree will not exceed rT nodes, this search strategy guarantees that, no more than rT subproblems are stored in the candidate list at any time. Rules (a) and (b) were implemented in the following manner to choose the separation variable x_{kq} .

i) First check rule (a). If it is applicable, then choose the variable x_{kq} with the largest period index (q) and the smallest product index (k). Otherwise go to step (ii).

ii) Use rule (b) to choose the variable with the largest period index and the smallest product index.

Again note that if $s_j = 0$ for all $j \in I$, then step (ii) does not apply.

We now briefly discuss how reoptimization was done in implementing the branch and bound algorithm for the case where we used linear underestimators. Let N^p and N^v be a pair of any parent and successor nodes, respectively. Also let B^p be the optimal basis for the linear program $(PR1^p)$. Going from node N^p to N^v exactly one row of matrix B^p changes. Let this new matrix be called B^v . There are two cases to consider.

Case a) B^v does not constitute a basis for the linear program $(PR1^v)$.

Case b) B^v is a basis for the linear program $(PR1^v)$.

In case (a) problem $(PR1^v)$ was solved from scratch. In case

(b), no reoptimization was necessary if B^v was in fact an optimal basis for $(PR1^v)$; primal (dual) simplex pivots were used to reoptimize if B^v was primal (dual) feasible but dual (primal) infeasible; and if B^v was neither primal nor dual feasible for $(PR1^v)$, both primal and dual pivots were used for reoptimization.

For the case when convex envelopes were used in constructing the subproblems, reoptimization is not as easy as in the case of linear underestimators. This is due to the fact that in this case more changes take place when one moves from a node to its successor. When at least one item for each family in each period was fixed at a positive production level then reoptimization for the convex envelope case became similar to that of the case of linear underestimator. Therefore we needed to obtain an initial solution at each node of the branch and bound algorithm. One possible alternative was to use the artificial variable method. Our preliminary study, however, showed that even the smallest problems could not be solved in a reasonable CPU time with this approach. This led us to search for a faster way of obtaining an initial basis. The matrix that corresponds to the coefficients of the constraints of a subproblem at any node of the branch and bound algorithm can be decomposed into two submatrices. One of these submatrices contains only the coefficients of the inventory balance equations and the second submatrix contains the coefficients of the underestimators of

the capacity constraints. Note that the former submatrix is the same at each node N^p of the branch and bound algorithm. First, we found the basis of a submatrix that corresponds to inventory balance equations. For this purpose we used the solution obtained from the heuristic procedure. A production variable x is assumed to be in the basis of the submatrix if its value is positive. Otherwise, an inventory variable y is assumed to be in the basis. We kept the inverse of this submatrix at each node of the tree. The basis of the second submatrix consists of the slack variables. The inverse of the original matrix is computed by using well-known principles of decomposition. This basis, however, is not necessarily feasible. Therefore, we use dual simplex iterations to obtain an initial feasible solution. Although this method is generally faster than the artificial variable method, as will be seen in the following section the effort required to find an initial feasible solution is a major disadvantage of the convex envelope approach.

5.3. Computational Experience with the Branch and Bound Algorithms For MFDLZ

As discussed earlier we developed two branch and bound algorithms for solving MFDLZ problems. One of the algorithms uses the linear underestimators and the other algorithm uses the convex envelopes to underestimate the capacity

constraints. As we will see in the following subsection test results of the latter algorithm are not encouraging. However, convex envelope results can still be used to obtain other underestimators for the subproblems. Hence we designed three experiments. Experiment 1 is the test of the branch and bound algorithm that uses linear underestimators in the subproblems. Experiment 2 is the test of the branch and bound algorithm whose subproblem at each node is obtained by using the convex envelope results. In experiment 3, we use the solutions obtained from the heuristic procedure and the convex envelope results to obtain a linear underestimator. This underestimator will be explained in section 5.3.3.

In all of the experiments, the data of the test problems were preprocessed by using only the 1st minimum alternative in the heuristic procedure.

In all of the experiments we used the same sets of problems. We randomly generated thirty sets of problems with varying dimensions. Each set contained five problems. In the first fifteen sets of problems individual setup time $s_j = 0$ for all $j \in I$ and the family setup time $S_i > 0$ for all $i \in \{1, 2, \dots, m\}$. These problems have prefix "A". In the second fifteen sets of problems both s_j and S_i values are positive. These problem sets have prefix "B". In generating these problems special attention was paid to make the problems "difficult". This was accomplished by making sure that the problems were "capacity-tight" and had several periods that

did not have enough capacity to produce all the demand. All problems have nonempty feasible regions.

For each problem, first a feasible solution, hence an initial upper bound was obtained with the heuristic procedure. Subsequently, using this upper bound each problem was subjected to preprocessing. The branch and bound algorithm was implemented on the preprocessed data with an initial upper bound. The heuristic, preprocessing and the branch and bound algorithm were all coded in FORTRAN and run on an IBM 3090/400 with vector processor.

5.3.1. Experiment One

The results of experiment 1 are shown in tables 4-7. We use the notation P.x.y.z in the first column of Tables 4-7. The first letter "P" indicates the prefix of the problem set and $x = r$, $y = T$, and $z = m$. The footnotes given in table 4 are also valid for tables 5, 6 and 7. In implementing the branch and bound algorithm for solving the test problems we adopted a tolerance to terminate the algorithm. The algorithm was terminated when we were assured that the current incumbent solution was within .2 % of optimality.

Computational results for the problem sets with prefix A are summarized in table 4. Let PERFIX denote the average percent of family setups fixed in the preprocessing step. For example for problem set A.4.6.2, $PERFIX = 8.6/12 = 71.7\%$.

For problem sets A we made the following observations.

- A-1. The average value of PERFIX for the fifteen problem sets was 46%.
- A-2. For fixed T and r, as m increases, PERFIX decreases and computational effort goes up. For example compare problem set A.6.6.2 to A.6.6.3 and problem set A.8.8.2 to A.8.8.4.
- A-3. For fixed r and m as T increases, PERFIX decreases and computational effort increases. For example compare problem set A.4.6.2 to A.4.8.2 and A.6.6.2 to A.6.6.3.
- A-4. For fixed T and m as r increases, so does PERFIX. In this case there is a reduction in the computational effort in general. For example compare sets A.4.6.2, A.6.6.2 and A.8.6.2, and sets A.4.10.2, A.6.10.2 and A.8.10.2.
- A-5. On the average an optimal solution used 95% of the total capacity over the T periods.
- A-6. Of the total capacity used for an optimal solution, 24% was used for setups.
- A-7. Average number of simplex(primal and dual) iterations required per linear program (subproblem) was 6.8.
- A-8. CPU time required to obtain an optimal solution ranged between 4.0-2760.6 seconds with an average of 90.0 seconds.

Computational results for the problem sets with prefix B are summarized in Table 5. For problem sets B we made the following observations.

- B-1. The average value of PERFIX for the fifteen problem sets was 54%.
- B-2. For fixed T and r as m increases, PERFIX decreases and so does the computational effort in general.
- B-3. For fixed r and m as T increases, PERFIX decreases and computational effort increases.
- B-4. For fixed T and m as r increases, so does PERFIX. But in this case computational effort increases.
- B-5. On the average an optimal solution used 93% of the total capacity over T periods.
- B-6. Of the total capacity used for an optimal solution 27% went to setups.
- B-7. It took on the average 3.4 simplex (primal and dual) iterations to solve each linear program.
- B-8. CPU time required to obtain an optimal solution ranged between 4.2-3150.0 seconds with an average of 267.8 seconds.

From Tables 4 and 5 and the observations listed above, it is evident that problems with zero (or negligible) individual setup times are easier to solve. This is a result of the fact that, problems with positive individual setup times have $T(m+r)$ sources of nonconvexity (and nonlinearity),

whereas in problems with zero individual setup times there are mT sources of nonconvexity (and nonlinearity). In other words, for problems with zero individual setup times each G_{it} , $i \in \{1, 2, \dots, m\}$ and $t \in \{1, 2, \dots, T\}$ is a source of nonconvexity and for problems with positive individual setup times, in addition to the G_{it} functions, we have each V_{jt} $j \in I$ and $t \in \{1, 2, \dots, T\}$ as a source of nonconvexity. The varying degrees of nonconvexity in these two problem types, translates into the following computational advantage in favor of the problems with zero setup times. For problems with $s_j = 0$ for all $j \in I$, if some variable x_{jp} , $p \in \{1, 2, \dots, T\}$, $j \in I_v$ and $v \in \{1, 2, \dots, m\}$ is declared positive at some node N^k , then there is no need to branch on any variable x_{qp} , $j \neq q$ and $q \in I_v$ at the nodes that succeed node N^k in the branch and bound tree. This clearly is not the case for problems with positive individual setup times. Consequently, for given T , the computational effort mainly depends on the number of families for problems with zero individual setup times and it depends mainly on the number of products for problems with positive individual setup times. This also explains why preprocessing is more helpful in reducing the computational effort for problems with zero individual setup times. With respect to the effect of preprocessing on the two types of problems, we remark that problems with zero individual setup times become easier to solve as the value of PERFIX increases. However, we are not able to draw a similar conclusion for problems with

positive individual setup times.

In the next part of our computational testing, we selected the last ten sets of problems from problem sets A and B and increased each period's capacity by 10% in each of the 100 problems. The problem sets thus obtained from sets A and B were prefixed D and E, respectively. The computational results with these problem sets are summarized in Tables 6 and 7, respectively.

Comparing Table 4 to Table 6 and Table 5 to Table 7 clearly indicates that problem sets D and E required much less computational effort than their respective counterparts. Observations similar to A-2, A-3, A-4 and A-6 for problem sets D and similar to B-2, B-3, B-4 and B-6 for problem sets E were made. In addition, the following points were observed for problem sets D and E.

- D,E-1. The average values of PERFIX were 79.7% and 81.1% for problem sets D and E respectively.
- D,E-5. On the average, an optimal solution used 87% of total capacity for problem sets D and 85% of total capacity for problem sets E.
- D,E-7. Average number of simplex iterations required per linear program was 6.2 for problem sets D and only 2.2 for problem sets E.
- D,E-8. CPU time required to obtain an optimal solution ranged between 4.2 - 133.8 seconds with an average of 10.6 seconds for problem sets D. For problem sets E the

range and the average of CPU seconds were 4.1 - 1180.2 and 119.1, respectively.

The considerable reduction in the computational effort when capacities were increased can be attributed to the following factors

i) The capacity constraints became "less binding" and as a result of this, subproblems in the branch and bound algorithm yielded feasible solutions more frequently. This in turn led to faster pruning of the branch and bound tree.

ii) Looser capacity constraints led to a reduction in computational effort required for reoptimization at the nodes of the branch and bound tree. This is evident from the smaller average number of simplex pivots required by sets D and E than their respective counterparts.

iii) The heuristic procedure yielded tighter upper bounds for sets D and E than for sets A and B, respectively. This in turn resulted in higher values of PERFIX in sets D and E, than in sets A and B, respectively.

In implementing the heuristic procedure, we generated one alternative by individual item release scheme, rather than generating r alternatives. The computational results of the modified heuristic procedure which includes all r alternatives are discussed in section 5.4. As we mentioned earlier, the heuristic procedure and preprocessing takes only a fraction of the total computational effort required for solving the test problems. The average CPU time required by the heuristic

Table 4. Test results of the branch and bound algorithm on 15 sets of MFDLZ problems with zero individual setup time (linear underestimator case.)

#	No. of LP's solved			No. of simplex iterations			No. of family setups fixed (a)			Ave. Cap. Used
	ave. (b)	min.	max.	ave. (b)	min.	max.	ave. (b)	min. (b)	max. (b)	
A.4.6.2	15.0	1	55	96.0	30	285	8.6	3	ALL	93.76
A.4.8.2	96.2	43	191	691.8	199	1709	5.2	3	9	96.47
A.6.6.2	7.4	1	21	104.6	44	196	9.6	6	ALL	94.80
A.6.6.3	43.0	7	89	306.18	144	563	6.8	2	12	93.00
A.6.8.2	61.4	11	169	802.4	118	2502	7.4	3	11	96.56
A.6.8.3	894.8	291	1647	5692.0	1609	12686	3.6	3	5	95.42
A.8.6.2	2.2	1	5	87.0	75	127	11.2	10	ALL	94.78
A.8.6.4	342.2	167	595	2327.8	767	3438	5.8	4	9	94.67
A.8.8.2	3.4	1	7	118.8	76	182	14.8	13	ALL	91.74
A.8.8.4	1691.0	351	4375	13021.8	2257	33172	6.6	4	12	94.73
A.4.10.2	251.4	37	513	1981.4	494	3728	4.8	2	8	95.52
A.6.10.2	64.2	3	133	728.8	111	1539	10.8	5	19	96.18
A.6.10.3	2505.8	751	3899	20263.8	5526	33056	3.4	3	5	94.92
A.8.10.2	53.8	1	195	998.6	106	3455	14.4	5	ALL	95.79
A.8.10.4	6079.4	2003	17901	38846.2	11986	110887	4.0	4	4	94.54

- (a) Number of family setups fixed indicates the minimum number of family setups that must be incurred in any optimal solution. For example, for a problem in the set A.6.6.3 there are a total of 18 setups over the 6 period planning horizon. Which specific families' setups are fixed in what specific periods are determined through preprocessing the problem data.
- (b) The averages are computed over the five problem in each set
- (c) Capacity usage for a given problem is the percent of total capacity used by the optimal solution.

Table 5. Test results of the branch and bound algorithm on 15 sets of MFDLZ problems with positive individual setup time (linear underestimator case)

Prob. #	No. of LP's solved			No. of simplex iterations			No. of family setups fixed			Ave. Cap. Used
	ave.	min.	max.	ave.	min.	max.	ave.	min.	max.	
B.4.6.2	119.0	29	217	309.6	105	654	8.6	4	ALL	91.09
B.4.8.2	538.2	271	983	1810.4	710	4083	5.6	2	10	92.31
B.6.6.2	423.4	135	1057	1053.4	262	2831	10.0	5	ALL	90.38
B.6.6.3	217.8	59	371	802.0	239	1967	11.0	4	16	92.05
B.6.8.2	2625.8	215	8673	6342.2	405	20072	11.2	8	ALL	92.86
B.6.8.3	3695.8	939	6137	15114.8	3851	25603	6.0	3	9	91.90
B.8.6.2	790.6	19	1465	2349.5	112	5435	4.8	11	ALL	93.62
B.8.6.4	2628.2	491	7113	9110.2	1247.4	30805	8.4	5	16	93.52
B.8.8.2	6048.2	373	18713	14693.4	1078	55872	13.4	10	ALL	93.65
B.8.8.4	8331.0	1463	20597	38948.7	7149	93267	8.4	5	15	93.50
B.4.10.2	4207.4	317	16443	14392.6	1285	46240	5.5	2	11	93.85
B.6.10.2	9331.6	121	21461	29411.8	543	83082	11.0	5	18	91.26
B.6.10.3	8553.0	531	17153	37805.7	2330	87617	9.0	3	16	92.99
B.8.10.2	25532.0	19635	31429	73744.0	56019	91469	17.0	15	19	93.76
B.8.10.4 ¹	24000.0	11541	36459	96250.5	62033	130468	5.0	4	6	91.93

1. Only two problems in this set could be solved in less than 3000 CPU seconds.

Table 6. Test results of the branch and bound algorithm on 15 sets of MFDLZ problems with zero individual setup time and 10% increase in capacity (linear underestimator case)

Prob. #	No. of LP's solved			No. of simplex iterations			No. of family setups fixed			Ave. Cap. Used
	ave.	min.	max.	ave.	min.	max.	ave.	min.	max.	
D.6.8.3	29.8	7	105	172.0	84	458	16.6	12	21	88.93
D.8.6.2	1.4	1	3	65.1	59	74	11.8	11	ALL	86.15
D.8.6.4	23.0	1	67	163.4	71	284	18.6	15	22	87.62
D.8.8.2	1.0	1	1	79.0	72	85	ALL	ALL	ALL	83.40
D.8.8.4	111.8	1	287	916.5	99	2373	21.0	5	ALL	88.56
D.4.10.2	5.8	1	15	85.0	56	144	17.4	14	ALL	89.23
D.6.10.2	2.6	1	5	85.6	61	108	18.8	17	ALL	88.80
D.6.10.3	83.8	7	263	601.2	100	1691	17.4	8	26	89.78
D.8.10.2	1.0	1	1	97.2	96	101	ALL	ALL	ALL	87.51
D.8.10.4	355.0	19	1343	1763.0	162	6318	19.0	5	31	89.17

Table 7. Test results of the branch and bound algorithm on 15 sets of MFDLZ problems with positive individual setup time and 10% increase in capacity (linear underestimator case)

Prob. #	No. of LP's solved			No. of simplex iterations			No. of family setups fixed			Ave. Cap. Used
	ave.	min.	max.	ave.	min.	max.	ave.	min.	max.	
E.6.8.3	254.6	49	819	609.4	172	1531	20.0	15	23	86.20
E.8.6.2	77.4	1	187	255.8	51	625	ALL	ALL	ALL	80.74
E.8.6.4	852.2	1	2429	1641.4	60	4015	20.4	17	23	85.55
E.8.8.2	1887.4	101	5127	3860.7	491	9365	ALL	ALL	ALL	85.43
E.8.8.4	1252.6	149	4999	3437.1	125	13745	27.4	23	31	86.38
E.4.10.2	672.2	59	1513	1479.9	239	3725	16.6	6	ALL	86.11
E.6.10.2	1503.8	359	4183	3534.1	949	11140	18.6	16	ALL	85.43
E.6.10.3	1809.4	115	3531	4270.9	272	7708	12.0	6	17	86.53
E.8.10.2	6994.2	67	16293	16313.1	298	42524	19.8	19	ALL	86.32
E.8.10.4	6727.0	229	15675	15559.1	792	37977	17.0	9	27	85.72

procedure for all the problems reported in Tables 4 through 7 ranged between .08-.11 CPU seconds with an average of .09 seconds. The heuristic procedure in all cases produced solutions that were within 0-23% of optimality. Furthermore, in more than 60% of all the problems solved, it produced an optimal solution. The CPU time required by the preprocessing step ranged between 1.90-23.40 CPU seconds with an average of 6.32 seconds. We also remark that whether or not $s_j = 0$ for all $j \in I$, does not have an impact on the computational effort required by the heuristic procedure and preprocessing. It is interesting to note that when we implemented the branch and bound algorithm on a subset of problems reported in Tables 4 through 7 without first running the heuristic and preprocessing the data, we observed a drastic surge in the computational effort. In some cases computational effort increased by a factor of more than 30.

The discussion presented in this section so far, indicates that, the branch and bound algorithm, together with the heuristic procedure and preprocessing, is more effective on MFDLZ when the individual setup times are negligible. Furthermore the branch and bound algorithm performs even better when the product families contain a relatively larger number of individual items. Therefore, it appears that the branch and bound algorithm would be a computationally feasible tool for solving MFDLZ, especially when individual setup times are negligible and when the product families include a

relatively large number of individual items. We have also demonstrated that MFDLZ problems with relatively less restrictive capacity constraints are much easier to solve. We are then led to conclude that for MFDLZ problems with larger dimensions ($r \geq 10$, $T \geq 10$), and nonnegligible individual setup times, and "tight" capacity constraints one has to rely mainly on heuristic procedures for obtaining "good" solutions.

5.3.2. Experiment 2

In experiment 2 we use the convex envelope of constraint (6) for the subproblems at the nodes of the branch and bound tree. As stated earlier the subproblems obtained in this way have more constraints and variables than the subproblems obtained by using linear underestimators. Another difficulty with the convex envelope approach is obtaining an initial basic feasible solution.

We solved 15 problems which were previously solved in experiment 1 and compared the results with the results obtained by using the branch and bound algorithm whose subproblems were constructed with linear underestimators (experiment 1). In all of these problems individual setup times were zero. We made the following observations.

1. Number of nodes stored in the convex envelope case is less than or equal to the number of nodes stored

in the linear underestimator case. This is due to the fact that the convex envelope is the tightest underestimator over the domain considered.

2. Number of linear programs solved in the convex envelope case is also less than or equal to the number of linear programs solved in the linear underestimator case. This is also because of the aforementioned fact.
3. In all but three of the problems solved number of simplex iterations in the convex envelope case is greater than the number of simplex iterations in the linear underestimator case. On the average, the former implementation requires 6.4 times as many simplex iterations as the latter. This difference is even bigger for larger sized problems. This is due to the fact that an initial basic feasible solution of the parent node could not be utilized in the former case. In other words, finding an initial basic feasible solution requires more computational effort. Also, high level of degeneracy in the convex envelope approach contributed to this result.
4. In three of the problems where convex envelope performed better in terms of the number of simplex iterations, lower bound obtained at node zero was equal to the upper bound and hence an optimal solution was obtained at the initial node of the

branch and bound tree.

5. In all problems, lower bound obtained at node zero with the convex envelope implementation is greater or equal to the lower bound obtained with linear underestimator implementation. The difference in the lower bounds, however, is not significant.

Although the convex envelope produces the tightest lower bounds the gain which can be obtained by using this property seems not to be worth when the additional effort for the convex envelope is considered. Since the problems selected for comparison were relatively easier problems to solve, we did not implement the convex envelope result for MFDLC problems.

5.3.3. Experiment 3

It is clear from experiment 2 that the computational effort required by the branch and bound algorithm that uses the convex envelope for the subproblems is rather prohibitive. However, when the number of linear programs and the number of nodes saved are considered it has a certain advantage over the linear underestimator. This fact led us to develop another underestimator which we call the convex relaxation. In this development we use the following property. The proof of property 8 is a direct consequence of theorem 5.

Property 8. For any $i \in \{1, 2, \dots, m\}$, $t \in \{1, 2, \dots, T\}$ and $x_t^i \in \Omega_{it}^2$ and $j \in \{1, 2, \dots, r_i\}$

$$H_{it}^j(x_t^i) = [(S_i + s_{\pi(1)}) / U_{\pi(1)t} + a_{\pi(1)}] x_{\pi(1)t} + \sum_{j=2}^{r_i} [s_{\pi(j)} / U_{\pi(j)t} + a_{\pi(j)}] x_{\pi(j)t} \leq r_{it}(x_t^i)$$

where for any $j \in \{1, 2, \dots, r_i\}$ $\{\pi(1), \pi(2), \dots, \pi(r_i)\}$ is a set of r_i subpermutations as given in definition 5. Note that the convex envelope is obtained by taking the maximum of $H_{it}^j(x_t^i)$ over all $j \in \{1, 2, \dots, r_i\}$. Therefore, we can use property 8 to develop a convex underestimator for the subproblems at each node of the branch and bound algorithm. The following lemma is an application of theorem 4.

Lemma 6. Let (x', y') be the solution to the heuristic procedure. Then for any $i \in \{1, 2, \dots, m\}$, any $t \in \{1, 2, \dots, T\}$

$$\max_{j \in \{1, 2, \dots, r_i\}} \{H_{it}^j(x_t^i)\} \leq r_{it}(x_t^i)$$

In constructing the convex relaxation for an underestimator we use lemma 6 and property 8 in the following manner. Let j' , $j' \in I_i$ for any $i \in \{1, 2, \dots, m\}$ be the product index for which lemma 5 is satisfied. By property 8, for each $i \in \{1, 2, \dots, m\}$, each $t \in \{1, 2, \dots, T\}$ and $x_t^i \in \Omega_{it}^2$ the convex relaxation is given by

$$H_{it}^{j'}(x_t^i) = [(S_i + s_{j'}) / U_{j't} + a_{j'}] x_{j't} + \sum_{\substack{j=1 \\ j \neq j'}}^{r_i} [s_j / U_{jt} + a_j] x_{jt} \quad (62)$$

In experiment 3, we use (62) as an underestimator for the subproblem at node N^0 of the branch and bound tree. Note that

the convex relaxation given in (62) is a linear function over the appropriate domain. As will be seen in the next section the solution obtained from the heuristic procedure is usually optimal and when it is not optimal it has an objective value that is very close to the optimal solution value. So the hyperplane which is obtained by using this solution is expected to be the maximum of the r_i hyperplanes and therefore, it is expected to give a tighter lower bound at node N^0 than the one obtained by using the linear underestimator.

We solved 25 problems whose individual setup times are zero (15 of these problems were the same problems of experiment 2) and compared the results of experiment 1 with those of experiment 2. We made the following observations.

1. For the problems where heuristic procedure was able to find an optimal solution the convex relaxation performed better both in terms of number of simplex iterations and the number of linear programs solved.
2. For small problems, number of linear programs solved, number of simplex iterations and CPU time in the convex relaxation case are similar to those in the linear underestimator case.
3. For some of the larger size problems the convex relaxation performed significantly better than the other two underestimators in terms of the number of simplex iterations.

4. For the problems where linear underestimator performed best, CPU time required by the convex relaxation approach is comparable to that required by the linear underestimator approach and in many cases the difference of the CPU times required is insignificant.

Based on these results we conclude that as the problem size gets bigger it is preferable to use the convex relaxation as an underestimator for the subproblems.

5.4. Computational Experience with the Branch and Bound Algorithm For MFDLC

To test the computational performance of the branch and bound algorithm for MFDLC we randomly generated thirty sets of problems with varying dimensions. Each set contained five problems. In the first fifteen sets of problems individual setup cost $k_j = 0$ for all $j \in I$, and individual setup times $s_j = 0$ for all $j \in I$. For all $i \in \{1, 2, \dots, m\}$ family setup cost and setup time $K_i > 0$ and $S_i > 0$, respectively. These problems have prefix "F." In the second fifteen sets of problems k_j , s_j , K_i and S_i values are positive. These problem sets have prefix "G." These problems were generated by using the same code developed for MFDLZ with minor modifications. Therefore, they are also "difficult" problems. Note that problem sets F are similar to problem sets A except the former problems

have family setup costs. Similarly problem sets G and B are alike except for the individual and family setup costs.

For each problem, first a feasible solution, hence an initial upper bound was obtained with the heuristic procedure. Using the upper bounds each problem is then subjected to preprocessing. The branch and bound algorithm was implemented on the preprocessed data.

Computational results with problem sets F and G are summarized in Tables 8 and 9, respectively. The footnotes given in table 4 are also valid for tables 8 and 9.

Although fifteen sets of problems with prefix F were generated, we were not able to solve the problems in the last two sets within 2000 CPU seconds. Therefore, we did not include them in table 8. Observations similar to A-2, A-3 and A-4 for problem sets F were made. In addition we made the following observations.

1. The average value of PERFIX for the thirteen sets of problems was 20%.
2. On the average an optimal solution used 94% of the total capacity over the T periods.
3. Average number of simplex (primal and dual) iterations required per linear program (subproblem) was 22.81.
4. CPU time required to obtain an optimal solution ranged between 6.40-1260.2 seconds with an average of 80.85 seconds.

Table 8. Test results of the branch and bound algorithm on 13 sets of MFDLC problems with zero individual setup time and setup cost (linear underestimator case)

Prob. #	No. of LP's solved			No. of simplex iterations			No. of family setups fixed			Ave. Cap. Used
	ave.	min.	max.	ave.	min.	max.	ave.	min.	max.	
F.4.6.2	315.4	113	765	1549.8	681	3436	2.8	2	4	90.61
F.4.8.2	958.6	275	1875	5642.0	1441	12157	2.6	2	5	95.46
F.6.6.2	46.2	39	55	571.0	417	742	3.0	2	4	89.59
F.6.6.3	217.4	197	241	2273.2	1998	2501	3.6	3	5	94.31
F.6.8.2	147.0	89	193	2408.2	1239	3035	2.4	2	3	96.81
F.6.8.3	2030.2	1091	4025	22213.4	13590	41799	3.0	3	3	93.68
F.8.6.2	17.0	9	25	350.6	268	542	6.2	4	8	93.04
F.8.6.4	1654.2	835	2675	20543.6	8967	35112	4.0	4	4	93.66
F.8.8.2 ¹	24.2	1	51	571.8	114	833	10.2	8	16	95.95
F.8.8.4 ¹	6166.0	4647	7685	85563.5	56541	114586	5.0	5	5	92.33
F.4.10.2	615.4	179	1029	5290.0	1236	8407	2.2	2	3	93.81
F.6.10.2	248.2	125	473	4237.8	2840	7234	2.4	2	3	91.20
F.6.10.3	1000.2	321	2337	16085.6	3552	37541	3.0	3	3	96.83

1. Only two problems in this set could be solved in less than 2000 CPU seconds.

Table 9. Test results of the branch and bound algorithm on 9 sets of MFDLC problems with positive individual setup time and setup cost (linear underestimator case)

Prob. #	No. of LP's solved			No. of simplex iterations			No. of family setups fixed			Ave. Cap. Used
	ave.	min.	max.	ave.	min.	max.	ave.	min.	max.	
G.4.6.2	209.0	95	539	1241.0	489	3480	2.4	2	3	92.36
G.4.8.2	1948.2	1011	4821	13017.4	4688	34976	2.2	2	3	94.75
G.6.6.2	1823.4	287	5105	11844.0	1498	30432	3.4	2	4	95.41
G.6.6.3	1175.4	545	2625	8047.2	4229	18526	3.2	3	3	93.27
G.6.8.2 ¹	4697.0	1541	7853	33585.5	13571	53600	3.5	3	4	94.13
G.6.8.3 ²	18943.0	18943	18943	145852.0	145852	145852	3	3	3	91.86
G.8.6.2	2359.2	257	5247	12803.8	2421	19934	5	3	8	91.88
G.8.6.4 ²	12883.0	12883	12883	87904.0	87904	87904	4	4	4	91.56
G.8.8.2 ¹	5087.0	1315	8859	52440.0	15049	89831	13.5	13	14	98.70

1. Only two problems in this set could be solved in less than 2000 CPU seconds.

2. Only one problem in this set could be solved in less than 2000 CPU seconds.

Computational results for the problem sets with prefix G are summarized in table 9. In these problem sets problems in the last six sets could not be solved with 2000 CPU seconds. Therefore, we did not include them in table 9. Observations similar to A-2, A-3 and A-4 for problem sets G were made. In addition we made the following observations.

1. The average value of PERFIX for the nine sets of problems was 27%.
2. On the average an optimal solution used 94% of the total capacity over the T periods.
3. Average number of simplex (primal and dual) iterations required per linear program (subproblem) was 7.47.
4. CPU time required to obtain an optimal solution ranged between 6.09-1320.02 seconds with an average of 170.47 seconds.

From tables 8 and 9 and the observations listed above, problems with negligible individual setup times and costs are easier to solve. This result concurs with the fact that, the problems with individual setup times and costs have more sources of nonconvexity. The same line of reasoning that was given in section 5.3.1 applies here. However, the incursion of setup costs makes the problem more difficult to solve. This fact can easily be seen when table 8 and table 9 are compared with tables 4 and 5, respectively. The increase in the level of difficulty can be attributed to the

following factors

i) The preprocessing scheme was not able to fix as many families as in experiment 1. Therefore, the branch and bound algorithm for MFDLC had to evaluate more subproblems.

ii) The heuristic procedure yielded tighter upper bounds for MFDLZ problems than for MFDLC problems.

iii) In the branch and bound algorithm for MFDLC, a node was fathomed only when lower bound to the subproblem was greater than or equal to the incumbent solution. Therefore, the depth of the branch and bound tree was more than the branch and bound tree in MFDLZ problems.

The discussion presented in this section so far, indicates that the branch and bound algorithm is not as effective as the one developed for MFDLZ. The branch and bound algorithm for MFDLC can not be used for practical problems unless better preprocessing schemes and heuristic procedures are developed for it. One has to rely on the heuristic procedures. Nevertheless, the test problems solved here can be used as benchmark problems against which the heuristic procedures can be evaluated.

5. 5. Computational Experience with the Heuristic Procedures for MFDLZ

The heuristic procedure for MFDLZ was tested on the problems that were solved optimally with the branch and bound algorithm. In implementing the heuristic procedure for MFDLZ,

we generated r alternatives using individual item release scheme as explained in section 4.4. We also generated alternatives using the family release scheme. First, we generated one shift alternative for each family in each period and hence generated m alternatives. Then we defined another set of families for each period as the combination of two families and generated one alternative for each such set. Therefore maximum number of alternatives generated in period t for each alternative of period $t+1$ is less than or equal to $(m^2 + 3m)/2$. For each family alternative we checked whether this alternative is inferior. We eliminated all such alternatives. Assume that for each period t , $t \in \{1, 2, \dots, T\}$ $K(t)$ alternatives are generated and for each of these alternatives there is a cost associated with it. Also assume that these alternatives are ordered in ascending order of their cost values. As discussed earlier we selected $k(t) \leq K(t)$, $t=1, 2, \dots, T$, alternatives. In the implementation, first $k(t)$, $t=1, 2, \dots, T$, was set to 10 and then it was set to 20. In both cases the heuristic procedure produced the same solution for all problems tested. The results of the heuristic procedure for the problem sets with prefixes A, B and E are reported in tables 10, 11 and 12, respectively. Heuristic procedure could generate a feasible solution to all problems. For the problem sets whose prefix is D, the solution obtained by the heuristic procedure was optimal for all problems. The average CPU time required to solve the

test problems is 3.01 seconds. The solutions to the 64%, 58% and 86% of the problems in set A, B and E, respectively were optimal. In 74% of the all problems, heuristic procedure obtained the optimal solutions. Average percentage difference between the solution of the heuristic procedure and the optimal solution is 1.16%, 1.91%, 0.08%, and 0.0%, for problem sets A, B, E and D, respectively. Overall average error bound of the solution of the heuristic procedure is 0.75%. In addition we made the following observations.

1. Computational effort required by the heuristic procedure ranges between 3.80-4.47 CPU seconds.
2. For fixed T and r , as m increases, percentage difference between the optimal solution and the heuristic solution increases. For example compare problem sets A.6.6.2 and A.6.6.3 in table 10, B.6.6.2 and B.6.6.3 in table 11, and E.6.6.2 and E.6.6.3 in table 12.
3. For fixed m and r , as T increases, percentage difference between the optimal solution and the heuristic solution increases. For example compare problem sets A.4.6.2 and A.4.8.2 in table 10, B.4.6.2 and B.4.8.2 in table 11, and E.8.8.4 and E.8.10.4 in table 12.
4. For fixed m and T , as r increases, percentage difference between the optimal solution and the heuristic solution does not change.

Table 10. Test results of the heuristic procedure on 15 sets of MFDLZ problems with zero individual setup time

Problem #	Difference between opt. and heuristic solutions (%)			No. of problems optimal solution found by heur.	No. of times 1st minimum alternative gives the best solution	Average CPU time in seconds
	ave.	min.	max.			
A.4.6.2	0.12	0.0	0.63	4	5	3.80
A.4.8.2	0.48	0.0	1.24	3	4	4.50
A.6.6.2	0.00	0.0	0.00	5	4	4.47
A.6.6.3	0.20	0.0	9.95	3	3	3.85
A.6.8.2	1.52	0.0	6.60	3	4	3.85
A.6.8.3	1.15	0.0	5.75	4	4	3.86
A.8.6.2	0.00	0.0	0.00	5	5	3.87
A.8.6.4	0.03	0.0	0.17	4	4	3.88
A.8.8.2	0.00	0.0	0.00	5	5	3.95
A.8.8.4	1.76	0.0	4.60	1	5	4.05
A.4.10.2	2.02	0.0	7.30	1	3	3.78
A.6.10.2	1.46	0.0	7.30	4	5	3.85
A.6.10.3	2.40	0.0	8.40	1	2	3.88
A.8.10.2	0.34	0.0	1.70	4	4	3.98
A.8.10.4	6.65	0.0	12.50	1	3	4.04

Table 11. Test results of the heuristic procedure on 15 sets of MFDLZ problems with positive individual setup time

Problem #	Difference between opt. and heuristic solutions (%)			No. of problems optimal solution found by heur.	No. of times 1st minimum alternative gives the best solution	Average CPU time in seconds
	ave.	min.	max.			
B.4.6.2	0.00	0.00	0.00	5	5	3.70
B.4.8.2	0.56	0.00	1.40	2	1	3.77
B.6.6.2	0.37	0.00	1.38	3	4	3.82
B.6.6.3	5.16	0.00	20.05	3	5	3.83
B.6.8.2	0.00	0.00	0.00	5	3	3.83
B.6.8.3	1.89	0.00	6.93	1	1	3.85
B.8.6.2	0.00	0.00	0.00	5	4	3.90
B.8.6.4	2.04	0.00	9.00	3	3	3.91
B.8.8.2	0.30	0.00	1.49	4	4	3.91
B.8.8.4	6.90	0.00	23.00	1	5	3.95
B.4.10.2	1.90	0.24	7.88	0	1	3.79
B.6.10.2	0.23	0.00	1.15	4	3	3.86
B.6.10.3	1.41	0.00	7.03	4	3	3.87
B.8.10.2	0.00	0.00	0.00	2	0	4.02
B.8.10.4 ¹	7.92	5.49	10.35	0	1	4.05

1. Optimal solutions of two problems are available in this set.

Table 12. Test results of the heuristic procedure on 10 sets of MFDLZ problems with positive individual setup time and 10% increase in capacity

Problem #	Difference between opt. and heuristic solutions (%)			No. of problems optimal solution found by heur.	No. of times 1st minimum alternative gives the best solution	Average CPU time in seconds
	ave.	min.	max.			
E.6.8.3	0.65	0.00	2.81	3	5	3.85
E.8.6.2	0.00	0.00	0.00	5	5	3.90
E.8.6.4	0.01	0.00	0.06	4	3	3.91
E.8.8.2	0.00	0.00	0.00	5	4	3.91
E.8.8.4	0.01	0.00	0.06	4	4	3.95
E.4.10.2	0.00	0.00	0.00	5	4	3.79
E.6.10.2	0.00	0.00	0.02	4	5	3.86
E.6.10.3	0.01	0.00	0.03	4	4	3.87
E.8.10.2	0.00	0.00	0.00	5	2	4.02
E.8.10.4	0.11	0.00	0.47	3	3	4.05

From tables 10, 11 and 12, and the observations listed above, it is evident that the heuristic procedure performs better as the problems get easier. However, for difficult problems whose optimal solutions are found with an excessive amount of computational effort the solutions found by the heuristic procedure are also reasonable.

Computational effort required by the heuristic procedure increases as the problem size gets bigger. The increase in the effort, however, is linear and thus does not impose any restriction for its practical usage.

In many manufacturing settings, capacity is not as tight as the capacity of the problems generated in our computational experiments. Although we do not have conclusive results for the performance of the heuristic procedure for realistically sized problems, the heuristic procedure can be used for all practical problems, since it performs very well in problems with loose capacity constraints.

5. 6. Computational Experience with the Heuristic Procedures for MFDLC

Heuristic procedure for MFDLC was tested on the problems whose optimal solutions were found by using the branch and bound algorithm. Generation of these problems was discussed in section 5.4. In implementing the algorithm, we used the same individual item release scheme and family release scheme

that were explained in section 5.5. Since the elimination of the inferior alternatives developed for the heuristic procedure for MFDLZ does not apply for this heuristic procedure, this routine is not included in the implementation. The procedure finds a starting solution by using the code developed by Erenguc, (1988). As discussed in the development of the heuristic procedure, the algorithm does not guarantee to find a feasible solution.

The results of the heuristic procedure for the problem sets with prefixes F and G are reported in tables 13 and 14, respectively. In one of the 150 problems that was solved, heuristic procedure could not generate a feasible solution. The solutions to the 45% and 29% of the problems in set F and G, respectively were optimal. In 40% of the all problems, heuristic procedure obtained an optimal solution. Average percentage difference between the solution of the heuristic procedure and the optimal solution is 9.11% and 4.06% for problem sets F and G, respectively. Overall average error bound of the solution of the heuristic procedure is 7.40%. In addition we made the following observations.

1. Computational effort required by the heuristic procedure ranges between 6.27-8.05 CPU seconds.
2. For fixed T and r, as m increases, percentage difference between the optimal solution and the heuristic solution increases. For example compare problem sets F.6.6.2 and F.6.6.3 in table 13, and

G.6.6.2 and G.6.6.3 in table 14.

3. For fixed m and r , as T increases, percentage difference between the optimal solution and the heuristic solution increases. For example compare problem sets F.4.6.2 and F.4.8.2 in table 13, and G.4.6.2 and G.4.8.2 in table 14.
4. For fixed T and m , as r increases, percentage difference between the optimal solution and the heuristic solution decreases. For example compare problem sets F.6.6.2 and F.8.6.2 in table 13, and G.6.6.2 and G.6.8.2 in table 14.

From tables 13 and 14, it is clear that the heuristic procedure for MFDLC is not as effective as the heuristic procedure developed for MFDLZ problems. However, as the capacity gets looser the procedure obtains a better solution. Since in many manufacturing environment capacity is not very tight, this heuristic procedure can be used for some real life problems. Nevertheless, this procedure is the first research effort in this direction.

Table 13. Test results of the heuristic procedure on 13 sets of MFDLC problems with zero individual setup time and cost

Problem #	Difference between opt. and heuristic solutions (%)			No. of problems optimal solution found by heur.	No. of times 1st minimum alternative gives the best solution	Average CPU time in seconds
	ave.	min.	max.			
F.4.6.2 ¹	7.91	0.00	25.59	1	3	6.27
F.4.8.2	10.73	0.02	19.47	0	3	6.69
F.6.6.2	3.75	0.00	9.15	2	3	6.57
F.6.6.3	6.45	0.00	17.80	3	4	7.13
F.6.8.2	1.79	0.00	4.71	2	4	7.32
F.6.8.3	14.33	2.84	34.77	0	2	7.40
F.8.6.2	0.00	0.00	0.00	5	5	6.99
F.8.6.4	8.21	0.00	10.99	1	3	7.54
F.8.8.2	0.00	0.0	0.00	5	4	7.99
F.8.8.4 ²	19.24	10.12	28.37	0	1	8.20
F.4.10.2	25.06	10.00	42.61	0	3	7.05
F.6.10.2	2.76	0.0	13.80	4	4	7.70
F.6.10.3	0.00	0.0	0.00	5	3	8.05

1. Heuristic procedure could not generate a feasible solution for one problem in this set.
2. Optimal solutions of two problems in this set are available.

Table 14. Test results of the heuristic procedure on 9 sets of MFDLC problems with positive individual setup time and cost

Problem #	Difference between opt. and heuristic solutions (%)			No. of problems optimal solution found by heur.	No. of times 1st minimum alternative gives the best solution	Average CPU time in seconds
	ave.	min.	max.			
G.4.6.2	0.53	0.00	2.17	3	4	6.42
G.4.8.2	4.50	0.00	19.91	1	5	6.80
G.6.6.2	3.75	0.00	9.15	0	4	6.97
G.6.6.3	5.17	0.00	10.12	1	5	7.01
G.6.8.2 ¹	0.00	0.00	0.00	2	0	7.29
G.6.8.3 ²	19.50	19.50	19.50	0	1	7.27
G.8.6.2	0.50	0.01	1.43	0	4	7.09
G.8.6.4 ²	0.00	0.00	0.00	1	1	7.58
G.8.8.2 ¹	2.94	0.50	5.37	0	2	7.84

1. Optimal solutions of two problems in this set are available.
2. Optimal solution of one problem in this set is available.

CHAPTER 6

SUMMARY EXTENSIONS AND FURTHER RESEARCH

In this dissertation we studied multi-family dynamic lot sizing models with coordinated replenishments. We presented certain properties of the models and developed an exact branch and bound algorithm for solving MFDLC. We developed two lower bounding schemes for use in the branch and bound algorithm. These are the linear underestimator and the convex envelope approach. We also developed a data preprocessing scheme for MFDLC. We also developed a heuristic procedure for MFDLC.

MFDLZ is a special case of MFDLC where setup costs are assumed to be negligible. Our branch and bound algorithm for MFDLZ is a modification of the branch and bound algorithm for MFDLC. We devised a heuristic procedure for MFDLZ which proved to be rather effective in providing a "good" upper bound for the optimal objective value of MFDLZ. We also developed a data processing scheme for MFDLZ which helped us obtain a tighter formulation of the problem.

We tested the branch and bound algorithm for MFDLZ on a set of 250 randomly generated problems. We used linear underestimators for the subproblems. Before implementing the branch and bound algorithm on these problems, we first obtained an initial feasible solution and an upper bound using

the heuristic procedure. The initial upper bound was used to preprocess the data. Preprocessing made it possible to fix certain family setups in certain periods. This of course resulted in a "tighter" problem formulation. Starting the branch and bound algorithm with an initial feasible solution and preprocessed data resulted in significant savings in computational effort. Of the 250 problems, 244 were solved for an exact optimal solution. Problems with negligible individual setup times were much easier to solve than those with positive individual setup times. Our computational results indicate that in those cases where individual setup times are negligible, capacity constraints are not "very tight" and each family contains a relatively large number of individual products, the branch and bound algorithm can solve realistically sized problems with a modest computational effort. We solved 15 problems out of these 250 problems by using the convex envelope in the subproblems. However, the computational effort required by the convex envelope approach was much more than that required by the linear underestimator approach. To obtain a better underestimating function we used convex envelope result and developed an underestimator which we called the convex relaxation. Our experience with the convex relaxation was very interesting. Computational effort with the convex relaxation is almost the same as the linear underestimator. Although in some cases linear underestimator seems to be better than the convex relaxation, as the problem

size becomes bigger the convex relaxation performs better. In any case, however, the future appears to be in the direction of developing effective heuristic procedures because of the computational burden of the branch and bound algorithms. We recognize the fact that the computational effort required by the branch and bound algorithm would be rather excessive especially for realistically sized problems with nonnegligible individual setup times. However, we would like to remark that we are able to provide a set of benchmark problems along with their optimal solutions against which future heuristics can be evaluated. We also note that if one is willing to accept a solution within for example 3% of the optimal solution, then the computational effort of the branch and bound algorithm may be drastically reduced.

Convex envelope gives the most accurate convex underestimating function over a certain domain. Convex envelope increases the number of constraints and the number of variables of the subproblems. Therefore, it may not be possible to implement the convex envelope results in the branch and bound algorithm. However, it can at least be used in the preprocessing step of the algorithm so that more items can be fixed before the branch and bound algorithm is started. The major difficulty with the convex envelope is the inclusion of additional constraints and variables. Posner and Wu (1978) developed a procedure for maximizing the minimum of a set of linear functions over a polyhedral set. Their procedure does

not introduce additional constraints for linearizing the objective function. The structure of the objective function given in Posner and Wu (1978) is similar to the additional constraints that are introduced in the convex envelope approach. It is worthwhile to explore the possibility of using their ideas in our problem.

The convex relaxation approach developed in chapter 5 appears to be comparable with that of linear underestimator approach. Some modifications over this approach could be made. One such possible modification is to use the current incumbent solution for developing the convex relaxation.

As it is clear from the computational results for MFDLZ, the branch and bound algorithm would require excessive computational effort for a realistically sized problems. It is obvious that MFDLC problems would require more computational effort than MFDLZ problems and therefore, their computational burden would be much more excessive. This fact is apparent from our experiments.

The computational results of the heuristic procedure for MFDLZ are rather encouraging. In almost 74% of the problems solved the heuristic procedure produced an optimal solution. In all cases the heuristic procedure produced solutions that were within 0-23% of optimality, the average being 0.75%. Our heuristic procedure can therefore be used for solving realistic size problems.

The computational results of the heuristic procedure for MFDLC are not as good as those of the computational results of the heuristic procedur for MFDLZ. The average gap between the solution of the heuristic procedure and the optimal solution is relatively high. Although this procedure is the first procedure developed for MFDLC, developing better heuristic procedures is in our future research agenda.

Lagrangean relaxation is one of the techniques that can be applied to solve integer programming problems (see for example Geoffrion, 1974; Fisher, 1981; Lasdon, 1970). Lagrangean relaxation was successfully applied to many integer programming problems (see for example Fisher, 1973; Fisher, 1985; Held and Karp, 1970; Shapiro, 1971; and Thizy and Wassenhove, 1985). Since MFDLC can be formulated as a mixed integer program, Lagrangean relaxation technique might produce tight lower bounds with less computational effort. It is therefore worthwhile to explore the Lagrangean relaxation technique for developing other algorithms for solving MFDLC and MFDLZ.

Another research issue that is awaiting the researchers is the decomposition of MFDLC. The computational burden of the branch and bound algorithms developed in this dissertation is mainly due to the length of the planning horizon. If some techniques can be developed to break the planning horizon into smaller horizons by preserving the structure of the problem, then the application of the branch and bound algorithms will be less prohibitive.

REFERENCES

- Aggarwal, S. C., "A review of current inventory theory and its applications," *International Journal of Production Research*, vol. 12, pp. 443-482, 1974.
- Aksoy, Y., and Erenguc S. S., "Coordinated multi-item lot sizing at a single stage: A survey," *International Journal of Operations and Production Management*, vol. 8, pp. 63-73, 1988.
- Avriel, M., "Methods for solving signomial and reverse convex programming problems," Avriel (editor): *Optimization and Design*, Prentice Hall, Englewood Cliffs, N.J., pp. 307-320, 1973.
- Avriel, M., and Williams, A. C., "Complementary geometric programming," *SIAM Journal of Applied Mathematics*, vol. 19, pp. 125-141, 1970.
- Bahl, H. C., "Column generation based heuristic for multi-item scheduling," *AIIE Transactions*, vol. 15, pp. 136-141, 1983.
- Bahl, H. C., Ritzman, L. P., and Gupta, J. N. D., "Determining lot sizes and resource requirements: A review," *Operations Research*, vol. 35, pp. 329-345, 1987.
- Barany, I, Van Roy, T. J., and Wolsey, L. A., "Strong formulations for multi-item capacitated lot sizing," *Management Science*, vol. 30, pp. 1255-1261, 1984.
- Benson, H. P., and Erenguc, S. S., "Using convex envelopes to solve interactive fixed charge linear programming problem," *Journal of Optimization Theory and Application*, vol. 59, pp. 223-246, 1988.
- Berry, W. L., and Mabert, V. A., "Research in production planning and inventory control: current trends and future directions," *AIIE Transactions*, vol. 13, pp. 100-101, 1981.
- Billington, P. J., "The capacitated multi-item dynamic lot sizing problem," *AIIE Transactions*, vol. 17, pp. 217-219, 1986.

Bitran, G. R., Haas, E. A., and Hax, A. C., "Hierarchical production planning: A single stage system," *Operations Research*, vol. 29, pp. 717-743, 1981.

Bitran, G. R., and Hax, A. C., "On the design of a hierarchical production planning systems," *Decision Science*, vol. 8, pp. 28-55, 1977.

Bitran, G. R., Magnanti, T. L., and Yanasse, H. H., "Approximation methods for the uncapacitated dynamic lot size problem," *Management Science*, vol. 30, pp. 1121-1140, 1984.

Bitran, G. R., and Matsau, H., "The multi-item capacitated lot size problem: Error bounds of Manne's formulations," *Management Science*, vol. 32, pp. 350-359, 1986.

Bitran, G. R., and Yanasse, H. H., "Computational complexity of the capacitated lot size problem," *Management Science*, vol. 28, pp. 1174-1186, 1982.

Brown, R. G., *Decision Rules for Inventory Management*, Holt, Rinehart and Winston, New York, 1967

Buffa, E. S., and Sarin, R. K., *Modern Production/Operations Management*, (8th edition), John Wiley, New York, 1983.

Clark, A. J., "An informal survey of multi-echelon inventory theory," *Naval Research Logistics Quarterly*, vol. 19, pp. 621-650, 1972.

Dixon, P. S., and Silver, E. A., "A heuristic solution procedure for the multi-item, single-level, limited capacity, lot sizing problem," *Journal of Operations Management*, vol. 2, pp. 23-40, 1981.

Dogramaci, A., Panayiotopoulos, J. C., and Adam, N. G., "The dynamic lot-sizing problem for multiple items under limited capacity," *AIIE Transactions*, vol. 13, pp. 294-326, 1981.

Doll, C. L., and Whybark, D. C., "An iterative procedure for the single machine multi-product lot scheduling problems," *Management science*, vol. 20, pp. 50-55, 1973.

Dzielinski, B., Baker, C. T., and Manne, A. S., "Simulation tests of lot size programming," *Management Science*, vol. 9, pp. 229-258, 1963.

Dzielinski, B., and Gomory, R., "Optimal programming of lot sizes, inventories and labor allocations," *Management Science*, vol. 11, pp. 874-890, 1965.

Eisenhut, P. S., "A dynamic lot sizing algorithm with capacity constraints," *AIIE Transactions*, vol. 7, pp. 170-176, 1975.

Eppen, G. D., and Martin, R. K., "Solving multi-item capacitated lot sizing problems using variable redefinition," *Operations Research*, vol. 35, pp. 832-848, 1987.

Erenguc, S. S., "Multiproduct dynamic lot-sizing model with coordinated replenishments," *Naval Research Logistics*, vol. 35, pp. 1-22, 1988.

Erenguc, S. S., Benson, H. P., "The interactive fixed charge linear programming problem," *Naval Research Logistics Quarterly*, vol. 33, pp. 157-177, 1986.

Falk, J. E., and Hoffman, K. L., "A successive underestimating method for concave minimization problems," *Mathematics of Operations Research*, vol. 1, pp. 251-259, 1976.

Falk, J. E., and Soland, R. M., "An algorithm for separable nonconvex programming problems," *Management Science*, vol. 15, pp. 550-559, 1969.

Fisher, M. L., "Optimal solution of scheduling problems using Lagrange multipliers: Part 1," *Operations Research*, vol. 21, pp. 1114-1127, 1973.

Fisher, M. L., "Lagrangian relaxation methods for solving integer programming problems," *Management Science*, vol. 27, pp. 1-18, 1981.

Fisher, M. L., "An application oriented guide to Lagrangian Relaxation," *Interfaces*, vol. 15, pp. 10-21, 1985.

Florian, M., Lenstra, J. K., and Rinnooy Kan, H. G., "Deterministic production planning: Algorithms and complexity," *Management Science*, vol. 26, pp. 12-20, 1980.

Fulop, J., "Reverse convex programming over a polytope," *Working Paper*, Budapest, 1988.

Gelders, L. F., and Van Wassenhove, L. N., "Production planning: A review," *European Journal of Operational Research*, vol. 7, pp. 101-110, 1981.

Geoffrion, A. M., "Lagrangian relaxation for integer programming," *Mathematical Programming Study*, vol. 2, pp. 82-114, 1974.

Goyal, S. K., "Determination of optimum packing frequency of items jointly replenished," *Management Science*, vol. 21, pp. 436-443, 1974.

Graves, S. C., "Using Lagrangean techniques to solve hierarchical production planning problems," *Management Science*, vol. 28, pp. 260-275, 1982.

Groover, M. P., *Automation Production Systems and Computer Aided Manufacturing*, Prentice-Hall, Englewood Cliffs, NJ, 1980.

Hanssman, F., "A survey of inventory theory from operations research viewpoint," R. L. Ackoff (editor): *Progress in Operations Research*, vol. 1, pp. 65-104, 1961.

Harris, F. W., *Operations and Cost*, In *Factory Management Series*, A. W. Shaw Co., Chicago, 1915.

Hax, A. C., and Candea, D., *Production and Inventory Management*, Prentice-Hall, Englewood Cliffs, NJ, 1984.

Hax, A. C., and Meal, H. C., "Hierarchical production planning and scheduling," *Studies in Management Sciences*, vol. 1, North-Holland, New York, pp. 53-69, 1978.

Held, M., and Karp, R. M., "The traveling salesman problem and minimum spanning trees," *Operations Research*, vol. 18, pp. 1138-1162, 1970.

Hillestad, R. J., and Jacobsen, S. E., "Reverse convex programming," *Applied Mathematics and Optimization*, vol. 6, pp. 63-78, 1980a.

Hillestad, R. J., and Jacobsen, S. E., "Linear programs with an additional reverse convex constraint," *Applied Mathematics and Optimization*, vol. 6, pp. 257-269, 1980b.

Horst, R., "Deterministic global optimization with partition sets whose feasibility is not known. Application to concave minimization, DC-programming, reverse convex constraints and Lipschitzian optimization," *Journal of Optimization Theory and Applications*, vol. 58, pp. 11-37, 1988a.

Horst, R., "Deterministic methods in constrained global optimization: Some recent advances and new fields of applications," *Working Paper*, University of Florida, 1988b.

Horst, R., and Dien, L. V., "A method for minimizing a DC-function subject to convex and reverse convex constraints," submitted, 1988.

Johnson, L. A., and Montgomery, D. C., *Operations Research in Production Planning, Scheduling and Inventory Control*, New York: John Wiley, New York, 1974.

Kao, E. P. C., "A multi-product dynamic lot size model with individual and joint setup costs," *Operations Research*, vol. 27, pp. 279-289, 1979.

Karni, R., and Roll, Y., "A heuristic algorithm for the multi-item lot-sizing problem with capacity constraints," *AIIE Transactions*, vol. 14, pp. 249-256, 1982.

Krajewski, L., King, B. E., Ritzman, L. P. and Wong, D. S., "Kanban MRP and shaping the production environment," *Management Science*, vol. 33, pp. 39-57, 1983.

Lambrecht, M. R., and Vanderveken, H., "Heuristic procedures for the single operation multi-item loading problem," *AIIE Transactions*, vol. 15, pp. 319-326, 1979.

Lasdon, L. S., and Terjung, R. C., "An efficient algorithm for multi-item scheduling," *Operations Research*, vol. 19, pp. 946-969, 1971.

Maes, J., and Wassenhove, L. N., "Multi-item single level capacitated dynamic lot sizing heuristics: A computational comparison (Part I: Static case)," *AIIE Transactions*, June 1986, pp. 114-123, 1986.

Magnasarian, O. L., *Nonlinear Programming*, McGraw-Hill, New York, 1969.

Manne, A. S., "Programming of economic lot sizes," *Management Science*, vol. 4, pp. 115-135, 1958.

Nahmias, S., "Inventory models," J. Belzer, A. G. Holzman and A. Kent (editors): *The Encyclopedia of Computer Science and Technology*, vol. 9, pp. 447, 1978.

Newson, E. F. P., "Multi-item lot size scheduling by heuristic part I: With fixed resources," *Management Science*, vol. 21, pp. 1186-1193, 1975.

Newson, E. F. P., and Kleindorfer, P. R., "A Lower bounding structure for lot size scheduling problems," *Operations Research*, vol. 23, pp. 299-311, 1975.

Peterson R., and Silver E. A., *Decision Systems for Inventory Management and Production Planning*, John Wiley, New York 1979.

Pinto, P. A., and Mabert, V. A., "Joint lot-sizing rule for fixed labor cost situation," *Decision Sciences*, vol. 17, pp. 139-150, 1986.

Posner, E. M., and Wu, C.-T., "Linear Max-min programming," Mathematical Programming, vol 20, pp 166-172, 1981.

Ritzman, L. P., King, B. E., and Krajewski, L. J., "Comparison of material requirements planning and reorder point systems," H. Bekiroglu (Editor): Simulation in Inventory and Production Control, Society for Computer Simulation, La Jolla, California, 1983.

Rosen, J. B., "Iterative solution of nonlinear optimal control problems," SIAM Journal on Control, vol. 5, pp. 223-244, 1966.

Shapiro, J. F., "Generalized Lagrange multipliers in integer programming," Operations Research, vol. 19, pp.68-76, 1971.

Silver, E. A., "A simple method of determining order quantities in joint replenishments under deterministic demand," Management Science, vol. 22, pp. 1351-1361, 1976.

Silver, E. A., "Coordinated replenishments of items under time-varying demand: Dynamic programming formulation," Naval Research Logistics Quarterly, vol. 26, pp. 141-151, 1979.

Silver, E. A., "Operations research in inventory management: A review and critique," Operations Research, vol 29, p 628-645, 1981.

Silver, E. A., and Meal, H. C., "A heuristic for selecting lot size quantities for the case of a deterministic time-varying demand rate and discrete opportunities for replenishment," Production and Inventory Management, vol. 14, pp. 64-74, 1973.

Spencer, M. S., "Scheduling components for group technology lines," Production and Inventory Management, vol. 21, pp. 43-49, 1980.

Sugimori, V., Kusunoki, K., Cho, F., Uchikawa, S., "Toyota production system and kanban system materialization of just-in-time and respect-for-human system," International Journal of Production Research, vol. 15, pp. 553-564, 1971.

Tersine, R. J., Principles of Inventory and Materials Management, (3rd edition), North-Holand, New York, 1988.

Thizy, J. M., and Wassenhove, L. N., "Lagrangian relaxation for the multi-item capacitated lot sizing problem: A heuristic implementation," AIIE Transactions, vol. 17, pp. 308-313, 1985.

Thoai, N. V., "On a class of global optimization problems," Discussion Paper, Institute of Mathematics, Hanoi, Vietnam, 1988.

Thompson, K., "MRPII in the repetitive manufacturing environment," *Production and Inventory Management*, vol. 23, pp. 1-14, 1983.

Thuong, T. V., and Tuy, H., "A finite algorithm for solving linear programs with an additional reverse convex constraint," *Lecture Notes in Economics and Mathematical systems*, vol. 225, Springer, pp. 291-302, 1984.

Tuy, H., "Convex programs with an additional reverse convex constraint," *Journal of Optimization Theory and Applications*, vol. 30, pp. 150-182, 1987.

Ueing, U., "A combinational method to compute the global solution of certain non-convex optimization problems," F. A. Lootsma (editor): *Numerical Methods for Nonlinear Optimization*, Academic Press, pp. 223-230, 1972.

Van Nunen, J. A. E. E., and Wessel, J., "Multi-item lot size determination and scheduling under capacity constraints," *European Journal of Operational Research*, vol. 2, pp. 36-41, 1978.

Van Roy, T. J., and Wolsey, L. A., "Solving mixed integer programming problems using automatic reformulation," *Operations Research*, vol. 35, pp. 45-57, 1987.

Veinott, A. F. Jr., "The status of mathematical inventory theory," *Management Science*, vol. 12, pp. 745, 1966.

Veinott, F. A. Jr., "Minimum concave cost solution of Leontief substitution models of multi-facility inventory systems," *Operations Research*, vol. 17, pp. 262-291, 1969.

Wagner, H. M., and Whitin, T. M., "Dynamic version of the economic lot size model," *Management Science*, vol. 5, pp. 89-96, 1958.

Whitin, T. M., "Inventory control research: a survey," *Management Science*, vol. 1, pp. 32-40, 1954.

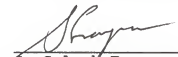
Zangwill, W. I., "A deterministic multiproduct, multifacility production and inventory model," *Operations Research*, vol. 14, pp. 486-507, 1966.

BIOGRAPHICAL SKETCH

H. Murat Mercan was born on January 14, 1959 in Agri, Turkey. He earned his bachelors degree in industrial engineering from Bogazici University in Istanbul in 1981. He received the Master of Science degree in industrial engineering at the same university in 1984.

Prior to his doctoral study Murat Mercan worked as a researcher in the Department of Industrial Engineering at Anadolu University, Eskisehir, Turkey. He was a graduate assistant in the Department of Decision and Information Sciences at the University of Florida between August 1984 and May 1989. Currently, he is a faculty member in the Department of Quantitative Business Analysis at Cleveland State University. He is a member of Decision Sciences Institute of America, The Institute of Management Science, Operations Research Society of America, and Production and Operations Management Society.


I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.


S. Selcuk Erenguc, Chairman
Associate Professor


I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.


Gary J. Koehler
Professor

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.


Harold P. Benson
Associate Professor

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.


Chung Y. Lee
Associate Professor

This dissertation was submitted to the Graduate Faculty of the Department of Decision and Information Sciences in the College of Business Administration and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

May, 1990

Dean, Graduate School